The Role of Symmetry Maps In Representing Objects In Images

by

Hüseyin Tek

B.S., Istanbul Technical University, 1989

M.S., Brown University, 1994

Thesis

Submitted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in the Division of Engineering at Brown University

PROVIDENCE, RHODE ISLAND

May, 2000

This dissertation by Hüseyin Tek is accepted in its present form by
the Division of Engineering as satisfying the
dissertation requirements for the degree of
Doctor of Philosophy

Date May 11, 00 [illegible]
Benjamin B. Kimia. Director
Division of Engineering

Recommended to the Graduate Council

Date 5/5/2000 David B. Cooper
David B. Cooper. Reader
Division of Engineering

Date May 4, 2000 [illegible]
David Mumford. Reader
Division of Applied Mathematics

Approved by the Graduate Council

Date 17 May 2000 [illegible]
Peder J. Estrup
Dean of The Graduate School and Research

Acknowledgments

I would like to thank Eileen M. Kane for sharing this long journey with me, for her continuous support and for the happiness that she gave me.

I sincerely thank my advisor Prof. Benjamin B. Kimia, for guiding me through this journey, for many insights, his boundless energy, his continuous encouragement and most of all for his friendship. I have learned a great deal, academically and otherwise and I truly enjoyed it. He has been understanding, supportive, generous, kind, and sympathetic.

I am grateful to Prof. David Cooper for his guidance, inspiration, advice and for his help on many occasions; I am also grateful to Prof. David Mumford for reading my thesis and making very valuable comments about this thesis. I would like to thank Prof. William A. Wolovich for his guidance during the early stage of my graduate studies. I would also like to thank my undergraduate teacher Prof. Fuat Anday for teaching me the joy of learning and research. He has had a great impact on my life both academically and otherwise.

I would like to thank Frederic Leymarie for introducing me to the CEDT algorithm and for many stimulating discussions. Thomas Sebastian was very helpful on many occasions. I would like to thank him for his endless help and for many valuable discussions. I would like to thank Dr. Kaleem Siddiqi for his help and support during my graduate studies. I would like to thank Perry A. Stoll for helping me with computer programs and for his friendship. I would like to thank Tolga Tasdizen for his friendship and his help at LEMS. I would like to thank George Mihalacopoulos and Dongjin Han for their friendship.

I would like to thank Michael M. Blane and John E. Adcock for their help with my computer related problems. They were great. I would like to thank Arpie Kaloustian, Cheryl Carvalho, and Ginny Novak for their help on many occasions. I would like to thank Kamil L. Ekinci, Hakan Civi, Nurdan Civi, Ömer Artun, Pınar Emirdag, and Burcak Pamir for helping me out in my difficult days.

I would like to thank Gülsen Özdamar for teaching me the joy of learning. I would like to thank Ediz Egeli for his friendship, for his honesty and for his positive effect on my life. I would like to thank the Kane family for their love, continuous support and for being my family in the United States.

This thesis is dedicated to my parents, Mustafa and Necibe, and my sister Fatma. I would like to thank them for their steadfast support and love.

And a special note of thanks and admiration for Matthew P. Kane, whose memory will always inspire me.

Contents

Chapter 1

Introduction

When we open our eyes, the forms in visual world surrounding us and their relative location are perceived clearly and instantaneously. One may therefore be led to believe that vision is a simple task because we are able to "see" the world without any special effort. However, this has has turned out to be misleading; the task of computational vision is indeed extremely complex. Three-dimensional physical structures project onto two-dimensional images and this loss of dimension has to be somehow reversed, *i.e.*, the physical structures must be inferred from these two-dimensional images, thus requiring careful assumptions about the properties of objects and their projections.

One of the main objectives of vision is to recognize 3D objects from the shape of their 2D projections. The object recovery and recognition approaches may be classified into *top-down* and *bottom-up*. On the one hand, top-down approaches search for and verify high-level models on the image [21, 19, 205, 239]. On the other hand, bottom-up algorithms require that shapes be robustly extracted from images, and represented in a manner that allows for robust correlation against a database of objects. This thesis is concerned with the recovery of shape (figure) from images and its representation in a fashion that allows for both bottom-up and top-down flow of information.

The task of figure/ground segregation is difficult primarily because a rich variety of visual transformations affect the projections of objects in two-dimensional retinal images. First, variations in the projections of the boundary and of the interior of objects change the precise contour-based or region-based description of object, yet often retain the qualitative shape. These variations come from a variety of sources including the noise which is added to gray-scale images by image acquisition devices, specularity highlights, cast shadows, edges from internal texture which are confused with apparent contours, as well as those induced by the limitations of low-level vision algorithms, Figure 1.1. Additionally, most low-level vision algorithms represent their result in the discrete domain, thus introducing unnecessary discretization errors, Figure 1.2. In an attempt to factoring out these variations a large class of methods aim to extract the coarse-scale structure of objects, either by smoothing the images or the resulting contours: some of these *smoothing* or *scale-space* methods are reviewed in Chapter 9. Second, occlusion and articulation rearrange or replace large chunks of objects due to changes in the view point. These variations keep the local description of some of the parts and some of the relative relationship between them unchanged, while affecting others. In general, three cases of occlusion can result from these variations. Specifically, (*i*) the most general case where both figure and occluder stand out, Figure 1.3a, and (*ii*) the occluder and background blend, introducing gaps in the representation, Figure 1.3b, and (*iii*) the occluder and the figure blend, leading to formation of parts, Figure 1.3c. A careful examination of the edge map in Figure 1.1 reveals examples of all these phenomenon. A large class of earlier methods aim to extract "parts" to establish invariance to the latter subclass of this set of visual transformations [158, 66, 195], Figure 1.3c, while some gap computation methods attempt to bridge across gaps [188, 4, 156, 78, 82, 234, 224]. These are reviewed in chapter 8. We will show that our symmetry representation and associated symmetry transforms present a language for decomposing all these operations (smoothing, partition, gap completion, removal of spurious edge elements) in a unified framework.

The top-down approaches, *e.g.*, [21, 19, 205, 239] are mostly used in specific applications, *e.g.*, road detection [21] and render object boundaries accurately. Specifically, the use of prior information about the objects that are searched in image domain makes these algorithm more stable to poor contrast boundary regions and occluded objects. However, the outcome of these approaches mostly depends on the extent of prior shape information used. The use of prior information however, limits these approaches to certain applications: it is difficult to build generic priors, although there are attempts in this direction [242, 142]. In addition, the use of application specific priors biases the visual system to perceive what it expects to perceive.

Previous bottom-up figure/ground segmentation algorithms can be classified into four groups [242]: (*i*) edge detection and grouping [32, 188, 4, 156, 78, 82, 234, 224], (*ii*) deformable models (snakes) [95, 49, 46, 48, 134, 212, 214], (*iii*) region growing [1, 243, 40], and (*iv*) global optimization of a functional [118, 142, 72, 45, 242]. These algorithms (*i*) often

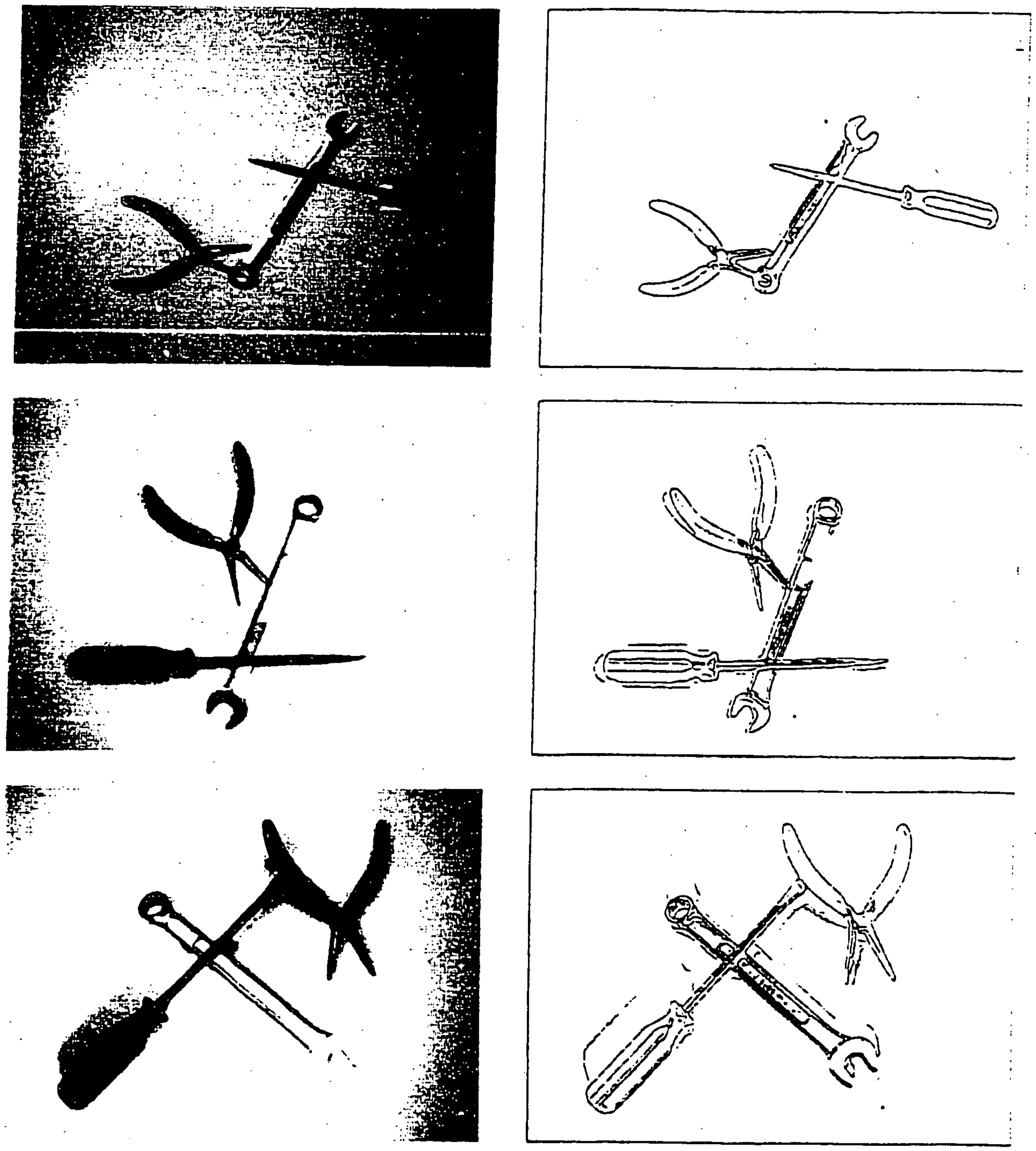


Figure 1.1: The outline of object boundaries are often distorted by photometric visual transformations and as well as by the limitations of low-level vision algorithms. This figure illustrates that the edge maps of the same objects acquired under different illumination conditions and relative configurations exhibit distortions from an ideal apparent contour, including gaps, occluded boundaries, spurious edges, and noise.

"jump" from low-level edge maps or (grouped) contour maps to complete object boundaries without taking advantage of an intermediate representation; (*ii*) separate segmentation and

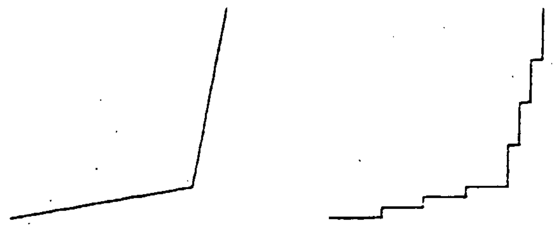


Figure 1.2: Representations of object boundaries are often distorted by quantization errors. Most low-level vision algorithms, but not all, output their result in the discrete domain, thus introducing unnecessary errors. The ability to represent the subpixel domain and operate on them reduces this addition discretization error.

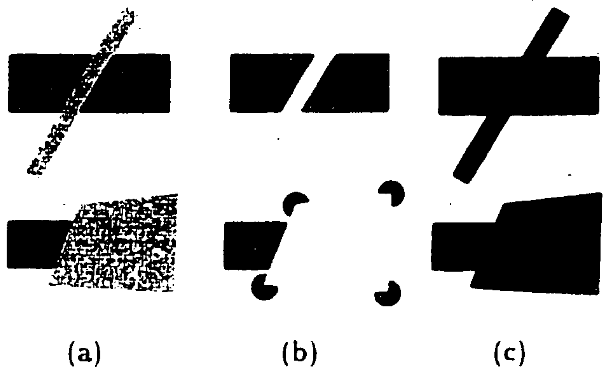


Figure 1.3: Examples of occlusion: the occluding object can (a) stand out from the background and the occluded object, (b) can blend with the background, thus creating *gaps*, or (c) blend with the object, thus creating *parts*.

recognition tasks into two distinct problems: and (*iii*) do not easily allow for top-down influence.

There is an increasing awareness that the division of object recovery task into segmentation and recognition stages can only be successful for a restricted application domain where the lighting conditions, viewing parameters, range of objects, *etc*. can be controlled. In this thesis, we propose that symmetry map (the medial axis augmented with a sense of dynamics and represented as a graph) can be used as an intermediate level representation mediating between low-level edge maps and high-level object representations. Specifically, the edge map of a real image can be fully represented by a symmetry map. These edges can then be perceptually organized by operating on the symmetry map. This allows for *both* bottom-up and top-down flows, and treats the segmentation and recognition aspects

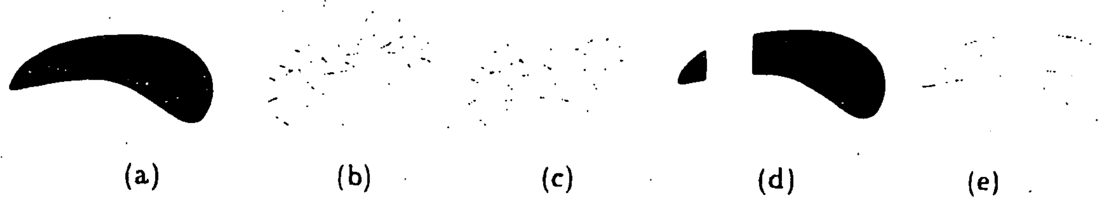


Figure 1.4: Shape continuity consists of both the continuity of the shape boundary as well as the continuity of its interior, which in effect implies joint continuity of a pair of boundaries. Compare the grouping process in (b) and (c). We represent both boundary and interior continuity with "skeletal continuity" which we propose as a measure to disambiguate edge maps.

of shape recovery as a single task.

1.1 Intermediate Level Representation: Symmetry Map

A key contribution of this thesis describes an algorithm for computing the symmetry map of an edge map on a discrete grid with no errors. The edge map is represented a collection of (possibly intersecting) free-form curve segments, each described as a geometric model. This approach combines analytic computations in the style of computational geometry and discrete propagations on a grid in the style of curve evolution and the numerical solutions of geometric PDE's such as those used in curve evolution. Specifically, waves from each of the initial curve segments are initialized and propagated as a discrete wavefront along discrete directions. In addition, to avoid error built up due to the discrete nature of propagation. shockwaves are detected and explicitly propagated along a secondary dynamic grid. The propagation of shockwaves. integrated with the propagation of the wavefront along discrete directions, leads to an exact simulation of propagation by the Eikonal equation. The resulting symmetries are simply the collection of shockwaves formed in this process which are recovered locally, exactly, and efficiently.

1.2 Perceptual Organization via Symmetry Transforms

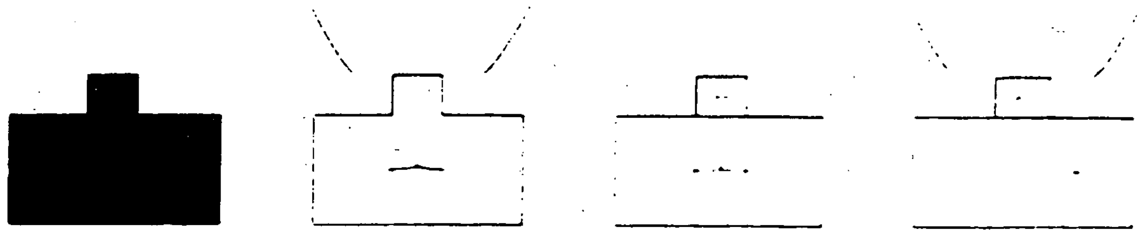
Low-level edge detectors lead to edge maps which are noisy and distorted, and which contain gaps and spurious elements as discussed earlier. The need to group edge elements to form object boundaries is faced with a range of ambiguities. Gestalt psychologists studied the nature of these ambiguities and proposed that among potential hypotheses for the organization of visual data, the human visual system selects those that depict regularities such as good continuation, proximity, symmetry, closure, *etc.* [168, 175]. These ideas have motivated an approach in computational vision where sets of edge elements which form long smooth contours with few interruptions are grouped together [188, 4, 156, 78, 82, 234, 224]. The smooth continuation of an edge element onto another is often an attribute of the best "completion curve" such as constructed from circular arcs [226], elastica [86, 141], stochastic elastica [141, 234] and close approximation of these [191]. The selection process as to which edge element is spurious and which one is to be grouped with others depends on a salience measure computed from the "goodness" of the "best" global curve as judged by its smoothness and length, as well the length and number of gaps.

Approaches that utilize the goodness of a global curve to determine the salience of edge elements and the subsequent grouping process have two major drawbacks. First, they do not take full advantage of "shape continuity" which depends not only on the continuity of each of the shape boundaries but also on the continuity of the shape interior as well. Figure 1.4. *Boundary continuity* depends on the individual regularity of each contour in isolation, while *shape continuity* is the *joint* continuation of a pair of contours. The latter is represented by shock/skeletal continuity, and requires both the shock geometry (skeletal locus) and the shock dynamics (flow of shocks along the skeleton) to be regular.

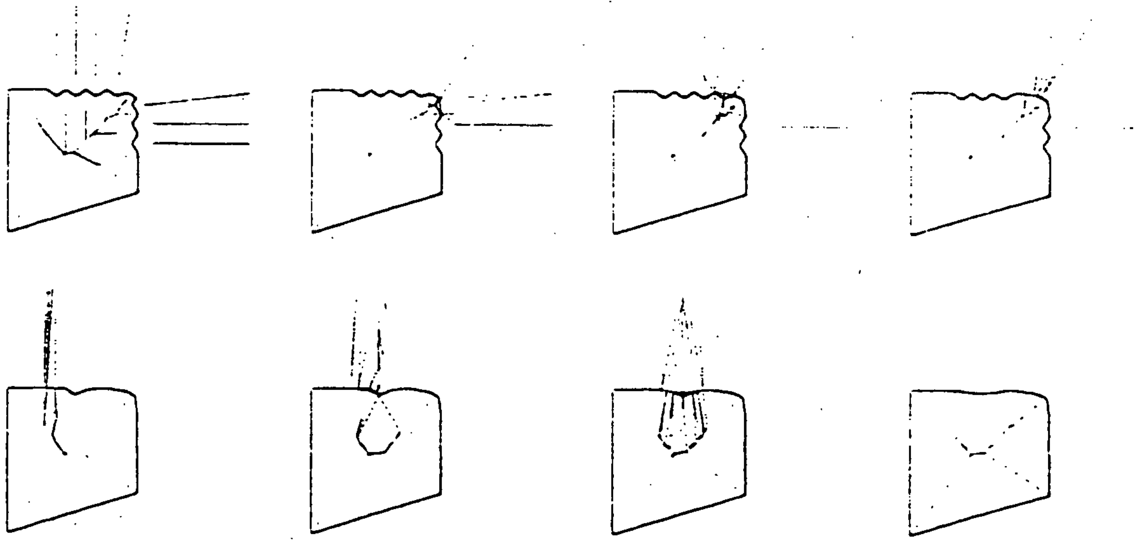
The second drawback involves the contribution of each edge element to competing hypotheses. Each edge element belongs to a single contour, and as such, its contribution to a hypothesis, should be in competition with its contribution to other competing hypotheses. In the psychophysics literature, each contour is perceived as belonging to a single object, namely, the occluder [76]. However, in most computational vision approaches edge elements contribute freely to competing hypotheses, mainly because the intermediate representations, *e.g.*, long smooth contours, have not been represented explicitly, nor is it immediately clear how to do so.

Our proposal for shape recovery relies on the explicit use of symmetries of an edge map as an intermediate visual representation. First, the symmetry map fully represents the initial edge map as well as any contour map resulting from grouping of edge elements [74]. Second, each visual transformation has a well-defined counter-part as a transformation on the symmetry map. Thus, grouping operations such as completing gaps, pruning spurious elements, as well as smoothing and partitioning the resulting shape, are expressible as a sequence of symmetry transforms on the initial symmetry map, Figures 1.5 and 1.6. Finally,

(a) gap completion via splice transforms



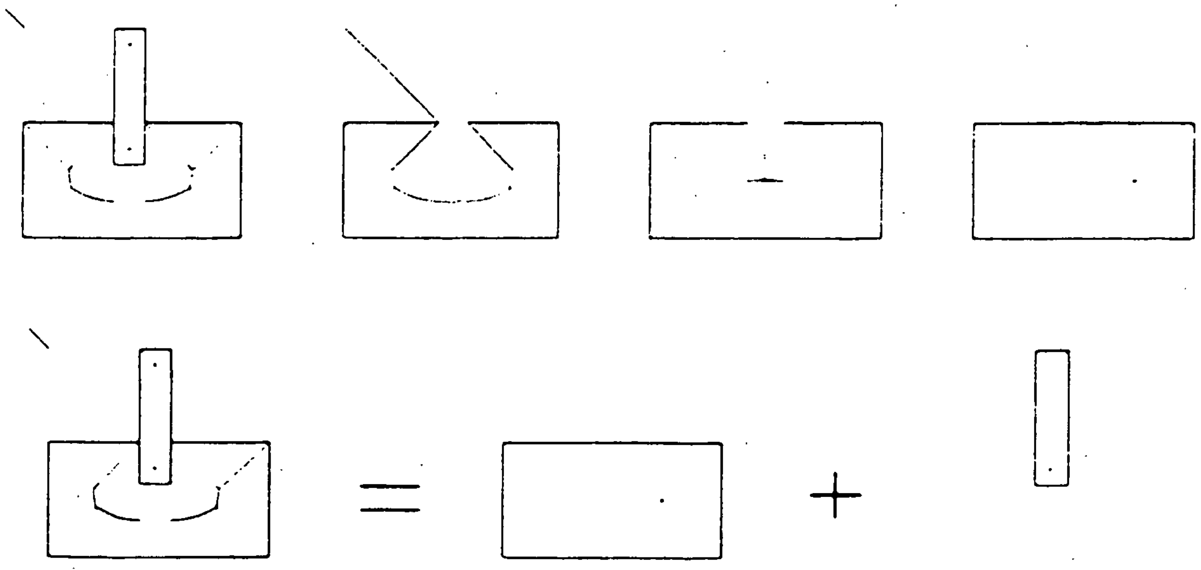
(b) part transform



(c) boundary smoothing via splice transform

⇒

(d) removal of spurious edge elements via loop transforms



(e) non-blending occlusion transform

Figure 1.5: This figure illustrates the main ideas behind the perceptual organization of edge maps via symmetry transforms. It should be noted that the contribution and focus of this chapter is the proposal that a symmetry map be used explicitly as an inter-meditate level representation between low-level edge maps and to show that all grouping operations are naturally expressible in this language. The task of selecting the optimal sequence is not addressed in this thesis.

(a)

(b)

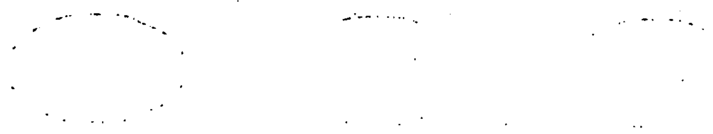


Figure 1.6: The application of a sequence of gap and splice transforms (not all steps are shown) [216]. This illustrates that perceptual grouping operations can be expressed in the language of a sequence of symmetry transform.

the optimal grouping corresponds to the least action path. or the sequence of symmetry transforms that best regularize the edge map. While we do not address the issue of how such an optimal sequence is selected here. (this is addressed in [225]).we emphasize that various hypotheses about objects in a scene are represented as sequence of transforms applied to the symmetry map. The contribution is the construction of a language to express such grouping hypotheses.

1.3 Objectives and Contributions of the Thesis

The main objective of this thesis is to recover shapes from images. Specifically. we propose a language for perceptual organization of edge elements via the notion of symmetry map of an edge map and symmetry transforms.

This thesis has three main contributions:

(1) We present an algorithm for detecting the symmetry maps of edge elements represented by free-form curve segments, *e.g.*, points, line segments, circular arcs, *etc.* This framework results in

(*i*) exact solutions

(*ii*) near optimal computational complexity (*iii*) local computations

(*iv*) graph representation for object recognition

(2) We present symmetry transforms operating on symmetry maps and resulting in symmetry maps which represent perceptual grouping operations *e.g.*,

(*i*) partitioning,

(*ii*) bridging gaps,

(*iii*) removing spurious edge elements,

(*iv*) dealing with occlusions.

(3) In the early stage of this thesis, I had developed a deformable model, the reaction-diffusion bubbles for image segmentation. This algorithm, which has been used in 3D medical image segmentation, *e.g.*, tumor detection, represents a wave propagation process which is not initialized from edges, but from the interior of objects, thus representing a dual to the above framework. This duality has led to the skeletally coupled deformable models, the thesis project of Thomas Sebastian of which I am a collaborator.

Part I

Symmetry Maps of Free-Form Curve Segments Via Wave Propagation

Chapter 2

Introduction

The goal of Part I of this thesis is to propose and develop the use of symmetry map as an intermediate level representation between low-level edge maps and high-level object representations. Specifically, it describes an algorithm for computing the symmetry map "exactly" [1] on a discrete grid for a collection of free-form curve segments each described as a geometric model. While the edge map is represented as a collection of (possibly intersecting) free-form curve segments, the symmetry map is represented as a planar, directed graph and is extracted by operations that are local, accurate (exact), and efficient. In addition, since grouping operations must be expressible on the symmetry map, the propagation must allow for the efficient addition to, modification of , or removal of geometric models from the edge map. Since the edge map contains numerous curve segments that are often not closed, or form junctions, existing methods for extracting skeletons are not always appropriate as detailed below, and a new framework for computing the symmetry map of an edge map is required.

The symmetry based representation of objects has received a great deal of interest not only in computer vision, but also in robotics, computational geometry, computer graphics, *etc.* Specifically, symmetry sets, also referred to as the skeleton, medial axis, Voronoi Diagram, *etc.* are used (*i*) as shape descriptors for object recognition [241, 101, 192, 199]; (*ii*) in simplifying (partitioning or grouping) visual forms [24, 2, 15, 219]; (*iii*) in image segmentation [230, 178]; (*iv*) image compression [29], (*v*) in path planning [16, 81], (*vi*) in statistical clustering [2] (*vii*) finite element meshing [204]; and (*viii*) in triangulating point sets [42] *etc.*

[1] exactness is limited by the computer's internal representation, *e.g.*, floating point accuracy.

Since Blum [24] proposed the "grass fire propagation" for symmetry detection, numerous symmetry detection techniques have been proposed. Methods for computing the medial axes (or skeletons), defined as the locus of maximally inscribed bitangent circles [24] may be classified in several categories: *Topological Thinning* [157, 91, 116], *distance transform methods* [53, 165, 71, 123], *curve evolution methods* [176, 112, 196, 209] and *computational geometry methods* [149, 16]. First, in topological thinning algorithms the pixels from the object boundaries are sequentely deleted whenever their removal does not alter the topology of the thinned shape. However, most thinning algorithms cannot always guarantee perfectly thinned shapes since there will be cases where the arrangement of pixels does not allow any further erosion. In addition, these methods are severely affected when the objects undergo rotation. Second, the distance transform methods involve three steps: (1) constructing the distance transform of a binary image [26, 53, 165], (2) extracting ridges or centers of maximal disks from the distance maps, (3) grouping (linking) these points into connected skeletons. Current distance transforms are not totally error free. In addition, robust extraction of their ridges from the distance transforms is numerically challenging, introducing further errors to the results, thus requiring post-processing. Third, curve evolution methods deform object boundaries by partial differential equations and the singularities (shocks) are then detected by detecting the discontinuities in the deforming surface [176, 112, 196, 209]. However, detecting singularities on evolving surfaces is a challenging task in the discrete domain, and requires a threshold to distinguish between significant singularities and noise. In addition, the notion of scale and detection are mixed in the process because some diffusion (smoothing) is introduced in the process for stable results. The main disadvantage, however, is that these methods embed a curve as the level set of a surface which a priori requires closed, non-intersecting curves and are thus not appropriate to deal with edge maps. Fourth, computational geometry methods analytically compute the "bisector" of two curves as the symmetry points from boundary pairs [149, 16], which produce rather well-connected and accurate results for discrete set of points, but require pruning [149]. For example, Voronoi Diagrams designed for a discrete set of points [16, 149] are very sensitive to quantization effects induced by the discretization and geometric transformations [149]. The approaches based on the curve evolution and computational geometry are reviewed in further detail in Chapter 7.

We propose a novel approach for detecting the symmetry map of an edge map of a grey-scale image [2]. The edge map is represented in the form of curve segments which in degenerate form also include points. The proposed framework is based on wave propagation which combines the curve evolution and computational geometry approaches. On the one hand, curve evolution methods are attractive because computations are local, topological events are handled properly, and accurate simulation methods are available. However, these approaches cannot operate on edge maps, are computationally expensive, and most importantly mix the detection and regularization stages of shock detection due to the errors produced evolution process. On the other hand, computational geometry approaches compute bisector curves of the curve segments representing the edge map. These bisectors, which are exactly equivalent to the the symmetry set of the curve segments lead to exact symmetries since computations are analytic. However, bisector computations are global and often require excessive calculations due to (i) the effort in detecting unnecessary symmetry branches and (ii) the effort in involved in removing them. The proposed framework [217] effectively combines these two approaches by developing an explicit wavefront propagation scheme on a *dual* grid: (i) discrete waves propagate curves segments on an *Eulerian* grid (fixed grid), which simulates the curve evolution paradigm: (ii) *shocks* (or singularities) which form during the propagation are explicitly represented and propagated on a *Lagrangian* grid (dynamic grid) constituting of the bisectors of the curve segments. This approach successfully combines the attractive properties of each approach, *i.e.*, the local computations of curve evolution and the exact results of bisector computations, while maintaining efficiency of computations. The results are in the form of planar, directed graphs which may then be used in object recognition without requiring any post-processing.

Specifically, we represent an edge map by N distinct geometric models, such as lines, circular arcs, conic sections, *etc.* each described by a set of parameters. A discrete wavefront is propagated from each source which eventually identifies one of the N element as the source of propagation for each discrete grid element (i, j). Clearly, a discrete grid cannot generally exactly represent the collision of fronts (singularities), nor the subsequent propagation of these singularities as shockwaves. However, the propagation of geometric models does allow for the analytic computation of the bisectors of each pair of sources. Since these bisectors form the path which singularities follow, we embed the intersection of bisectors with gridlines as a secondary (Lagrangian) discrete grid superimposed on the initial fixed (Eulerian) discrete grid, and propagate a discrete wavefront which consists of point along both grids.

[2]if the edge map is a closed curve it results in the traditional medial axis.

The addition of this dynamic grid leads to exact results which are computed efficiently. Finally, the addition or removal of a geometric model involves the propagation of new waves from this element *without* having to recompute the entire symmetry map.

In this part, we will present a symmetry detection algorithm based on a wave propagation algorithm as follows. In Chapter 3 we will review the mathematical construction of wavefront propagation based on the Huygens' and Fermat's principles, and study the analytic solutions of the Eikonal equation which models propagation of waves. In Chapter 4 we will address the numerical solution of the wave propagation from a set of sources, *e.g.*, points, lines, circular arcs, and more generally, any geometric models which can be represented by a finite set of parameters. Specifically, previous approaches to this problem will be reviewed and a novel discrete wavefront propagation algorithm will be presented. It will be shown that the discrete domain solutions to the wave propagation problem are not totally-error free because wavefronts may not be represented on a discrete grid when *shocks* form during the wave propagation. In Chapter 5, we will show that these errors could be avoided if shockwaves that form from the collision of wavefronts during the wave propagation process are represented explicitly, thus allowing exact wavefront propagation. We will illustrate how these shockwaves can be efficiently and accurately detected by using ideas borrowed from computational geometry. In Chapter 6, we describe this simultaneous discrete wavefront and shockwave propagation algorithm in detail. In addition, several examples will be given to illustrate this shock detection algorithm. In Chapter 7, we will review previous symmetry detection approaches based on curve evolution and Voronoi Diagrams and draw comparisons since our approach is based on the combination of analytic curve evolution and bisector computations.

Chapter 3

Wavefront Propagation: The Analytic Solutions

In this chapter, we review the construction of wavefront propagation based on the Huygens' and Fermat's principles. These two equivalent, but complementary, views dominate the traditional models of wave propagation. On the one hand, Huygens' principle asserts that an evolving wavefront is the envelope of wavelets emanated from each points of an earlier wavefront. On the other hand, Fermat's principle states that waves must travel along paths of shortest time. Both formulations lead to the Eikonal equation

$$u_x^2 + u_y^2 = \frac{1}{c^2(x,y)}, \tag{3.1}$$

where $u(x,y) = t$ describes the wavefront at time t, and $c(x,y)$ is the speed of the wavefront. The general solutions are typically found by the *method of characteristics*. These general solutions can be computed up to the time when characteristics (or rays) collide with each other. When characteristics collide with each other shocks (or singularities) form and flow on the *shock paths*. In general, determining the location of a shock path is a difficult task, both analytically and numerically. Approximating solutions to the Eikonal equation can then be obtained by entropy satisfying numerical methods. However, when the speed of the wavefront is constant, *i.e.*, $c(x,y) = constant$, the solution to the Eikonal equation constructs the distance transform of the initial data and the shock paths correspond to the equi-distance points from the initial sources, thus making the exact solutions to the Eikonal equation possible.

3.1 Construction of Wavefront Propagation

In this section, we review the derivation of a partial differential equation for wave propagation that is the basic model in geometrical optics [51, 97, 129, 23, 93]. Mainly, we consider the propagation of an optical disturbance in an isotropic medium. Specifically, let us assume that a disturbance at certain time $u(x, y) = u_0 = constant$ at the point $P = (x, y)$ propagates locally in an isotropic manner with speed $c = c(x, y)$. The disturbance then spreads along a circular ring of radius $c_0 \Delta u$, at time $u_o + \Delta u$, Figure 3.1a. Specifically every point on the wave front surface now emits a disturbance with a propagation speed $c(x, y)$, which depends on the location of the disturbance. The disturbances emitted from the sequence of points P_1, P_2, P_3 lying on the surface will move to P_1', P_2', P_3'. The wave front at time $u + \Delta u$ will be the envelope of all these disks, Figure 3.1. This geometrical construction is known as Huygens's principle.

Let us now construct an analytical description of the wavefront surface, $u(x, y)$. The light rays are curves orthogonal to the wavefront at each point and constitute, Figure 3.1. Let us denote the infinitesimal displacement vector along a light ray by ds, $ds = (dx, dy)$. From the definition of increment, du, we have

$$du = u_x dx + u_y dy = \nabla u . ds = |\nabla u||ds| \tag{3.2}$$

since ds is orthogonal to $u(x, y)$. Since du denotes "time increment"

$$du = \frac{|ds|}{c} \tag{3.3}$$

Thus, eliminating ds from (3.2) and (3.3) gives

$$|\nabla u|^2 = \frac{1}{c^2} \tag{3.4}$$

or

$$u_x^2 + u_y^2 = \frac{1}{c^2} \tag{3.5}$$

This equation is called the *Eikonal equation*. In this equation, the surface, $u(x, y)$ measures the time at which the wavefront crosses, or visits the point (x, y).

Another approach to construct the analytic description of wave propagation is based on the Fermat's principle, which states that waves travel along the paths of least action, *i.e.*, shortest time. Specifically, let us consider two fixed points $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ and assume that the speed of the light is $c(x, y)$. The travel time for the light is given by

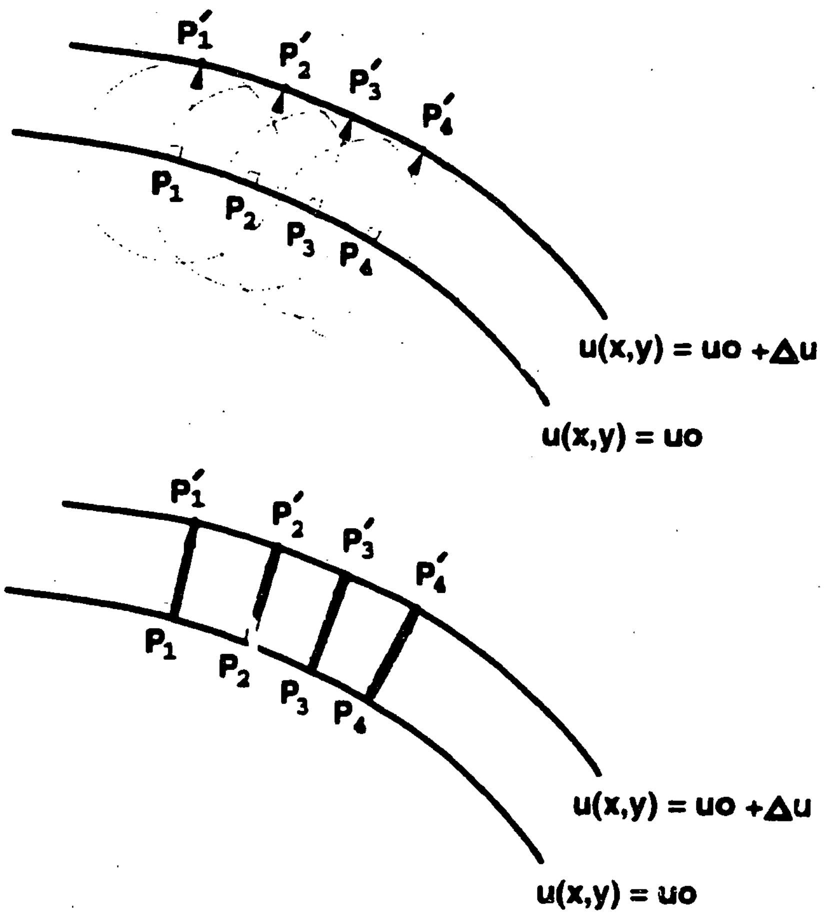


Figure 3.1: Two complementary views of propagation: Huygen's principle views each point as a source and the new wavefront as the envelope of each source's propagation front, while Fermat's principle focuses on the propagation of rays from each point along the least action paths.

the line integral [97]

$$u = u_Q - u_P = \int_P^Q \frac{ds}{c(x,y)}, \tag{3.6}$$

and the minimum time path is obtained by minimizing u by a variational principle. This minimization in fact leads to the Eikonal equation [97, 27], *i.e.*, $u_x^2 + u_y^2 = \frac{1}{c^2}$.

While this Eikonal equation is cast into a stationary PDE, *i.e.*. $u(x,y)$ is independent of time, interestingly, the propagation of wavefronts has also been represented by an evolutionary partial differential equation. Specifically, suppose that a curve $C(s,t) = (x(s,t), y(s,t))$ represents a wavefront at time t. In addition, consider propagation of a wavefront $C(s,t)$ along its normal direction, Figure 3.2. According to the Huygen's construction, the solution at time t, $C(s,t)$ can be constructed by the envelope generated by the set of all disk of radii

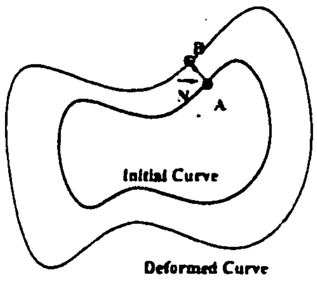


Figure 3.2: The points on the initial curve A move to B to generate a new curve.

$c(x,y)dt$ centered on the initial curve $C(s,t=0) = C_0(s)$:

$$\frac{\partial}{\partial t}C(s,t) = c(x,y)\vec{N}(s,t) \tag{3.7}$$

where $c(x,y)$ is the speed of the front and $\vec{N}(s,t)$ is the unit normal vector of the curve $C(s,t)$ [182, 99]. This approach is referred to *curve evolution* and has been extensively used in crystal growth [187], flame propagation [181], and shape representation [99, 106].

In this section, we have presented two equivalent but complementary approaches to the wave propagation problem. In the next section, we will study the general solutions of the Eikonal equation by using the *method of characteristics.*

3.2 The General Solution of The Eikonal Equation

The method of characteristics tracks "particle", *i.e.*, points which maintain their "identity" in space and time. These trajectories can be computed by rewriting the Eikonal equation with $c(x,y) = c_0 = contant$ in terms of two new variables p and q defined as

$$\begin{cases} p = u_x \\ q = u_y \\ p^2 + q^2 = 1/c_0^2 \end{cases} \tag{3.8}$$

or writing it in short form as

$$F(x,y,u,p,q) = p^2 + q^2 - 1/c_0^2 = 0 \tag{3.9}$$

Suppose that solution $u(x,y)$ is known along a curve Γ_0 given by

$$x = x_0(s), \quad y = y_0(s), \quad u = u_0(s), \tag{3.10}$$

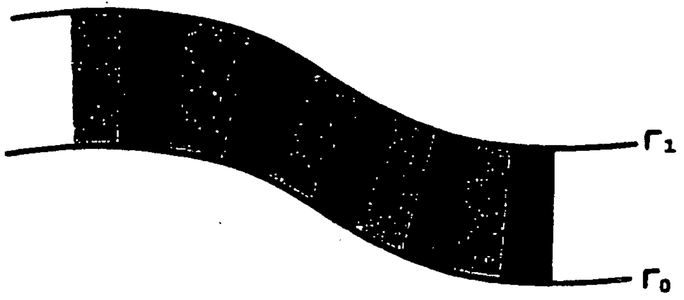


Figure 3.3: This figure illustrates characteristic strip expansion for the construction of the solution, Γ_1 to a nonlinear PDE from the given initial curve, Γ_0.

The solution $u(x,y)$ to this system along a new curve Γ_1, which is in the neighborhood of the initial curve Γ_0 can be computed by following the *characteristic strip* defined by the method of characteristics [23, 97, 51]. Specifically, the method of characteristics converts the nonlinear PDE (3.9) into a system of ordinary differential equations [23, 97]:

$$\begin{cases} \frac{dx}{ds} = F_p = 2p \\ \frac{dy}{ds} = F_q = 2q \\ \frac{du}{ds} = pF_p + qF_q = 2/c_0^2 \end{cases} \tag{3.11}$$

and

$$\begin{cases} \frac{dp}{ds} = -F_x - pF_u = 0 \\ \frac{dq}{ds} = -F_y - qF_u = 0 \end{cases} \tag{3.12}$$

A solution to Equations (3.11) and (3.12) along which $F(x,y,u,p,q) = 0$ is called a *characteristic strip*. These five equations define a characteristic strip which smoothly joins Γ_0 to Γ_1, Figure 3.3.

The first two equations state that the base characteristics, or the light rays are orthogonal to the wavefronts, *i.e.*, the rays are tangent to the surface normal $(p,q) = (u_x, u_y)$. Note that this is even true when the speed of waves are not constant, $c(x,y) \neq$ *constant*. From Equations (3.12) and (3.11) we also conclude that rays are straight lines for $c(x,y) = c_0$. However, when $c(x,y) \neq$ *constant* this is no longer valid since $\frac{dp}{ds} = -\frac{c_x}{c^2(x,y)}$ and $\frac{dp}{ds} = -\frac{c_y}{c^2(x,y)}$. The solution to these equations are then given by

$$\begin{cases} x(s) = x_0(\tau) + 2p_0 s \\ y(s) = y_0(\tau) + 2q_0 s \\ u(s) = u_0(\tau) + \frac{2}{c_o^2} s \\ p_0^2(\tau) + q_0^2(\tau) = 1/c_o^2 \end{cases} \tag{3.13}$$

It should be emphasized that the functions defined in the initial strip (3.10) are not entirely arbitrary [97]. They must: (i) be consistent with (3.9) - that is $F(x_0, y_0, u_0, p_0, q_0) = 0$. and ($ii$) satisfy the strip condition, that is, $\frac{\partial u_0}{\partial s} = p_0(\tau)\frac{\partial x_0}{\partial s} + q_0(\tau)\frac{\partial y_0}{\partial s}$. The initial values of p_0 and q_0 are then obtained from these two conditions. It is now straightforward to solve s and τ in terms of x and y and to substitute these values in the expression for u. If there is a solution to (3.8) at point τ_0, then a sufficient condition for this solution in the neighborhood of this point is that

$$J = F_p y_0' - F_q x_0' = p_0 y_o' - q_0 x_0' \tag{3.14}$$

is not equal to zero. This ensures that the base characteristics (or rays) are not parallel to the base initial curves defined by Equation (3.12). Eliminating p_0. q_0. and s from Equation 3.13. the general solution to (3.8) is then given by

$$(x - x_0)^2 + (y - y_0)^2 = (u - u_0)^2 c_0^2. \tag{3.15}$$

In general, two definitions connect the general solution $u(x, t)$ and the initial boundary $u_0(x, 0) = u_0(x)$:

Range of Influence: A point x_0 on the initial boundary can only influence a region of the solution domain. This region is called *the range of influence* of a point x_0, Figure 3.4a. This range for a point on a regular curve can be detected by moving along the characteristics from that initial point up to the places where characteristics intersect with each other.

Domain of Dependence: The *domain of dependence* of a point (x^*, t) is defined as the set of initial points that effect the solution there, Figure 3.4b. These points can be determined by going back to the boundary along the characteristics from the point (x^*, t).

3.2.1 Special Case: Distance Functions

Let us now consider a special case where $c(x, y) = 1$. In this case, the Eikonal equation is

$$u_x^2 + u_y^2 = 1, \tag{3.16}$$

which is the differential equation of the straight light rays. Suppose that the initial surface $u(x_0, y_0) = 0$ at time $t = t_0$. In this case, the value of function $u(x, y)$ at any point (x, y) in

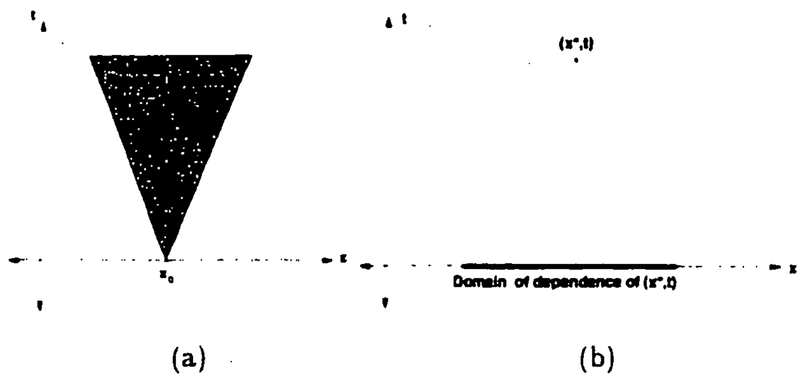


Figure 3.4: This figure sketches (a) the range of influence of a point x_0 on the initial wavefront; (b) the domain of dependence of point (x^*, t) on the solution domain.

the plane is equal to the *distance* of this point from the initial curve $u(x_0, y_0) = 0$.

$$u(x, y) = \sqrt{(x - x_0)^2 + (y - y_0)^2}. \tag{3.17}$$

In this model, $u(x, y) = constant$ represents the wavefront in the x-y plane and the characteristics are the light rays. The curves $u(x, y) = t$ are equidistant and parallel to the curve $u(x, y) = 0$.

3.2.2 The Domain of the General Solutions

The general (or classical) solution to the Eikonal equation can be constructed by the method of characteristics as shown above. Specifically, when $J \neq 0$ the initial value problem has one and only one solution to the Eikonal problem. What happens when $J = 0$? First, if $J = 0$ everywhere along the initial curve, there are an infinite number of solutions if the initial curve is a characteristic curve itself. Second if $J = 0$ everywhere along the non-characteristics initial curve, it is not possible to compute a continuously differentiable solution by the characteristic method. In this case, the characteristics cross each other and $u(x, y)$ can no longer be defined as a single-valued function of x and y.

Let us now present an example illustrating some of the ideas presented above. We showed that the rays emanating from the initial curve propagate orthogonal to the wavefront. Let us now consider the analytic wavefront propagation as moving along the characteristics (rays). Figure 3.5 illustrates the propagation of rays (red) and the the wavefront (blue) from a parabola. Specifically, the rays propagate (shown only for the inward inward direction) pass through a cusped *caustic*, which is the evolute (envelope of the rays, *i.e.*, the locus of centers of curvature) of the initial wavefront. Similarly, once the wavefronts reach the

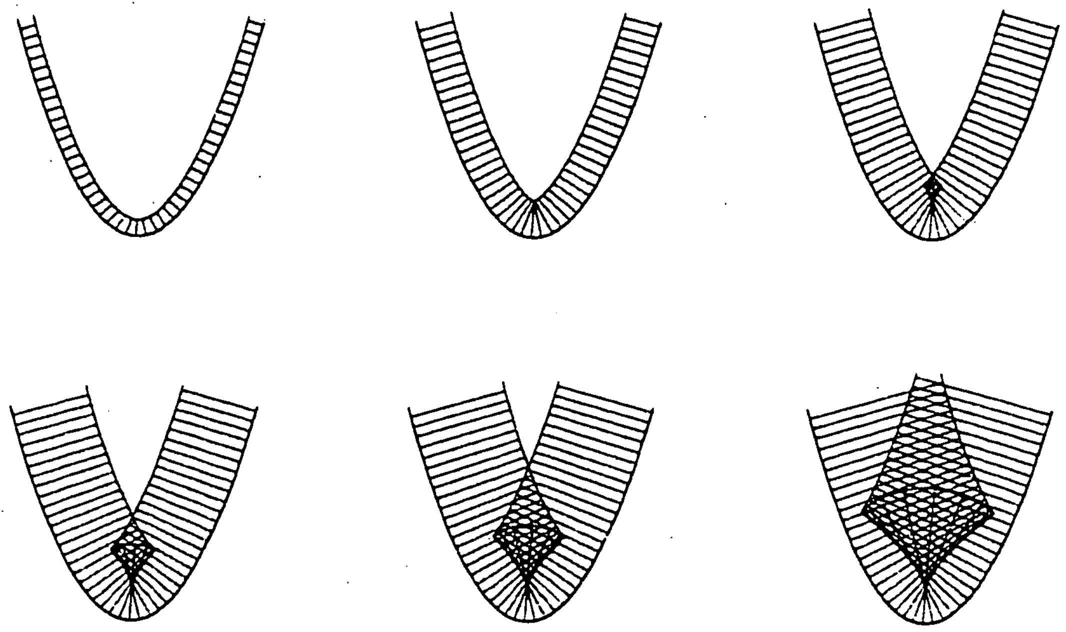


Figure 3.5: This figure illustrates the propagation of rays from an initial source shaped as a parabola. Observe that when rays collide, they cross each, thus forming a cusped caustic. Inside the caustic, the solution surface forms a *swallowtail* and becomes multiple (three) valued.

cusp point, the wavefront creases and becomes three-valued inside the caustics. Note that below the envelope curves, $u(x,y)$ is well-defined, whereas along this caustic $J(s,\tau) = 0$ and no singular solution to the characteristic equation is available afterwards. Since in most physical situations a triple valued point does not make sense, once characteristics cross each other, there is no classical solution of the PDE representing the wavefront propagation, due to multi-valuedness. Instead, a single valued *generalized solution*, also referred to as a *weak solution* is sought when the characteristics intersect with each other.

In addition, when the normal to initial data cannot be defined, such as at end points, or a discontinuity of a source, a range of normals are used, leading to the formation of rarefaction waves [203]:

Definition 1 *Regular wavefronts are those points of the wavefronts which arise from contour points with regular tangents.* Rarefaction *or* degenerate *wavefronts arise from points having no or multiply defined tangents.*

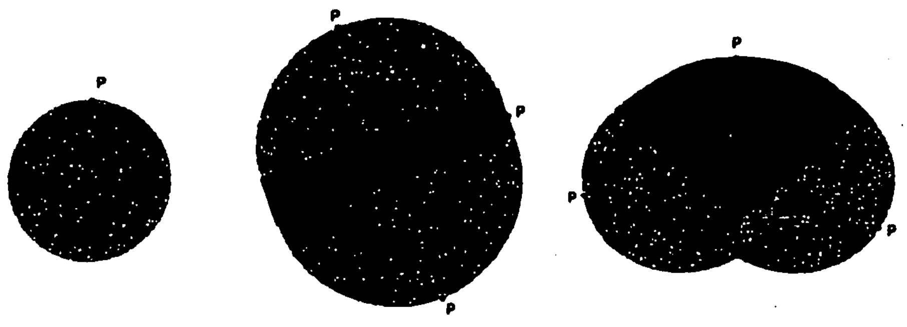


Figure 3.6: Examples of propagation of regular (dark shade) and rarefaction waves (light shade) for a point, a line segment, and a circular arc.

3.3 Discontinuous Solutions Of Conservation Laws

When characteristics intersect with each other, a weak solution, where a discontinuity in u(x,y) is allowed, can be constructed by preventing characteristics from crossing each other. Specifically, a discontinuity, or shock, forms when the characteristics collide and consequently propagates on a path which separates the quenching wavefronts. Figure 3.7 now illustrates that the rays originated from a parabola terminate at a curve which corresponds to a discontinuity of the solution $u(x,y)$. The discontinuous solutions can be obtained by viscosity solutions, which allow discontinuities in the solution.

3.3.1 Vanishing Viscosity Solutions

Most physical processes represented by hyperbolic PDE's ignore the effects of various dispersive effects which are modeled by higher order derivatives which are multiplied by small coefficients called *viscosity coefficients*. *Vanishing viscosity* approaches try to obtain solutions for the hyperbolic equations as the limits of solutions of some parabolic equations as the viscosity coefficients go to zero. Let us now consider *level set evolution* [154] to illustrate the vanishing viscosity solution. Specifically, the evolution of a surface, $\phi(x,y,t)$ by constant speed, c_0, is given by [154]

$$\phi_t(x,y,t) + c_0|\nabla\phi(x,y,t)| = 0 \tag{3.18}$$

where the zero level set of $\phi(x,y,t)$ represents an evolving curve, $C(x,y,t)$. The solution, $\phi(x,y,t)$ develops discontinuities when the normals collide during the evolution [154]. In-

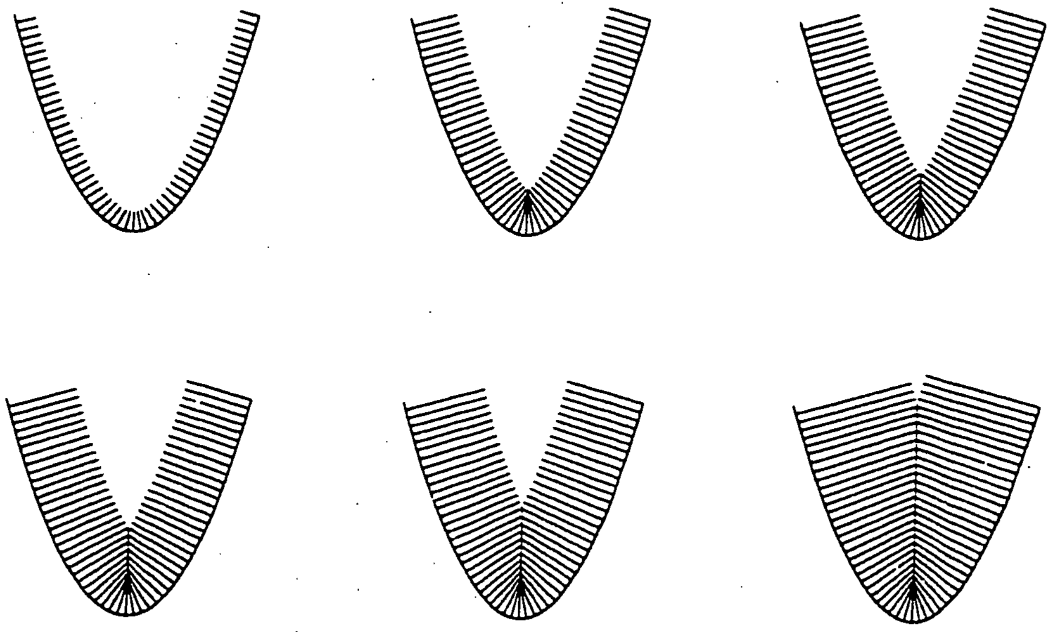


Figure 3.7: When the rays from a parabola collide, a discontinuous solution is sought to prevent rays crossing each.

stead, let us consider the following parabolic equation which contains a diffusive term

$$\phi^{\epsilon}_t(x,y,t) + c_0|\nabla\phi^{\epsilon}(x,y,t)| = \epsilon\kappa|\nabla\phi^{\epsilon}(x,y,t)| \tag{3.19}$$

where $\kappa = \frac{\phi_{xx}\phi_y^2 - 2\phi_{xy}\phi_x\phi_y + \phi_{yy}\phi_x^2}{(\phi_x^2+\phi_y^2)^{3/2}}$ is the curvature of the evolving surface $\phi^{\epsilon}(x,y,t)$. Suppose that the initial curve is a parabola as illustrated in Figure 3.8. It is well-known that the solutions of parabolic equations are smooth even when the initial data is not smooth. Figure 3.8 illustrates evolution of the curve represented by the zero level set of the surface $\phi^{\epsilon}(x,y,t)$ inward for the varying values of ϵ. Specifically, when ϵ is large the high curvature locations are smoothed faster than the other points. When the *epsilon* approaches to zero the effects of smoothing becomes negligible. Thus, the discontinuous solution $u(x,y,t)$ can be obtain as the limit of these smooth solution as ϵ goes to zero - that is

$$\phi(x,y,t) = lim_{\epsilon\to 0}\phi^{\epsilon}(x,y,t) \tag{3.20}$$

However, this vanishing viscosity method is not practical. Instead, weak solutions derived from conservation laws and the entropy conditions are used for the viscosity solutions. Chapter 4

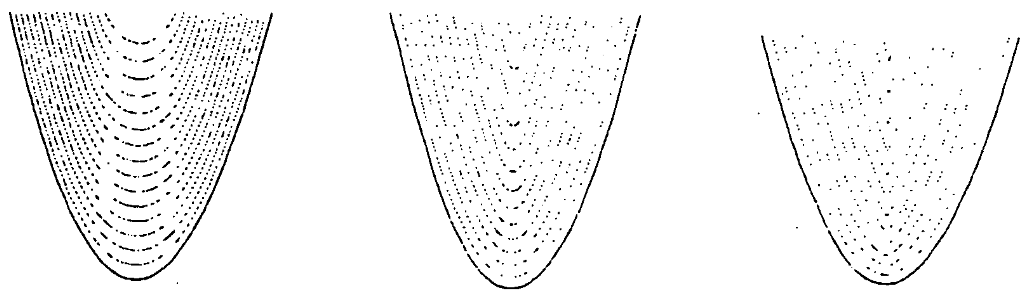


Figure 3.8: This figure illustrates the vanishing viscosity approach for the parabolic equation (3.20). (a) $\epsilon = 0.3$ (b) $\epsilon = 0.05$, and $\epsilon = 0.001$

3.3.2 Exact Solutions To The Eikonal Equation

The general solution to the hyperbolic equations computed by the characteristic method cannot be globally defined because the solutions do not remain smooth in general, even when the initial condition is smooth. Specifically, the general solutions can be obtained up to the time when the shockwaves form from the quenching of characteristics. When a shockwave forms it propagates along a shock path that is formed by the intersecting rays. Thus, the *exact solutions* to a hyperbolic PDE can be obtained if the shock path, which separates region where the general solutions holds, is computed. In general, determining the space-time location of the shock path is a difficult task, both analytically and numerically. Approximating solutions to the hyperbolic PDE's can be obtained by entropy satisfying numerical methods.

Let us return to the wavefront propagation which is analytically represented by the Eikonal equation, $\nabla u(x,y) = \frac{1}{c^2(x,y)}$. If the waves propagate with constant speed, *i.e.*, $c(x,y) = c_0$, the solution $u(x,y)$ can be computed by the distance function inside each influence zone, whose boundaries are defined by shock paths. Thus, the exact solution to this specific PDE can be obtained if the shock paths of the solution can be computed analytically.

Suppose that the constant speed waves are initiated at the same time from boundaries. The rays representing these waves quench with each other at places where the quenching rays have the same solution value. Thus, the shock paths can be analytically computed by finding places which are equi-distant from two or more boundary sources. Specifically, suppose that a shock path separates two wavefronts, $u_n(x, y)$ and $u_m(x, y)$. This shock path, s(x,y) can be computed by solving

$$u_n(x, y) - u_m(x. y) = 0, \tag{3.21}$$

where $u_n(x, y)$ measures the distance from the boundary source. n. These equi-distance paths. or *bisectors* can often be analytically computed if the boundary is represented by a known geometric model such as a point, a line segment. a circular arc, a conic arc. *etc.* When this is not possible numerical methods can be obtained [166].

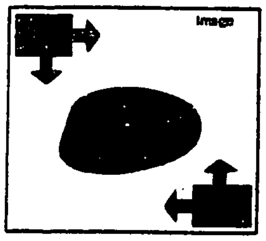
Chapter 4

Wave Propagation: Discrete Solutions

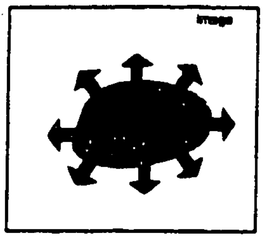
In the previous chapter, we have reviewed the classical solutions to the Eikonal equation

$$u_x^2 + u_y^2 = \frac{1}{c^2} \tag{4.1}$$

which models the propagation of waves and showed the critical role of the shockwaves in obtaining solutions to it. In this chapter, we will specifically focus on the discrete domain solutions. We will first review previous approaches to generate the discrete distance transform since when $c(x, y) = contstant$, the solution to the Eikonal constructs the distance transform from the initial source. Second, we will review some of the numerical methods for solving the Eikonal equation, namely, explicit front propagation via normal vectors, level set method and the fast marching algorithm. Third, we will present the discrete wavefront propagation algorithm which is based on an intermediate view of the Huygens' principle and the Fermat's principle. Two ideas play an important role in the design of the algorithm: (i) the rays are orthogonal to initial boundary models, *i.e.*, fixed directional operators; and (ii) the solution (wave equation solution) at a grid location is computed by measuring the distance between the grid location and the source of the wavefront, *i.e.*, geometric boundary model. Fourth, we will show that these the discrete domain solutions to the Eikonal equation are not totally error free, without representing the singularities, thus motivating simultaneous propagation of shocks, Chapter 5.



(a)



(b)

Figure 4.1: (a) This figure illustrates how sequential distance transforms are constructed by scanning the image with masks two (or more). (b) Distance transforms can also be constructed by applying the masks *iteratively* on the each location of the object boundaries, thus propagating the distance information towards inward and outward.

4.1 Discrete Distance Transforms

The *Distance Transform* of an image containing binary objects is an image where the value at each location (x, y) represents the closest distance from the objects in the corresponding image. Traditionally, the object boundaries in images have been represented by the discrete set of points due to the simplicity and computational issues. Thus, most distance transforms take advantage of simplicity of the discrete boundary representation.

Distance transforms may be classified into groups by the metric that they use. The most common and intuitive metric in distance transforms is the Euclidean metric [53], due to its scaling and rotational invariance properties. However, Euclidean distance transforms of discrete images are known to have significant computational complexity. Thus, several different techniques based on the approximations to the Euclidean metric [170, 26, 59] are proposed to ease the computational burden. These metrics between two points $x = (x_1, x_2)$ and $y = (y_1, y_2)$ can be summarized as

(i) $d(x, y) = max_{i=1,2}(|x_i - y_i|)$; the *chess-board* metric

(ii) $d(x, y) = |x_1 - y_1] + |x_2 - y_2|$; the *city-block* metric

(iii) $d(x, y) = max(|x|, |y|)a + min(|x|, |y|)(b - a)$, the *chamfer* metric

where a ad b are fixed parameters. Chamfer metric has been the most popular of them due to the its close approximation the Euclidean metric. One of the major goals in the design of the Chamfer distance transforms is to reduce the error between computed distance and the corresponding the Euclidean distances by finding the optimal selection of a, b parameters [26, 31].

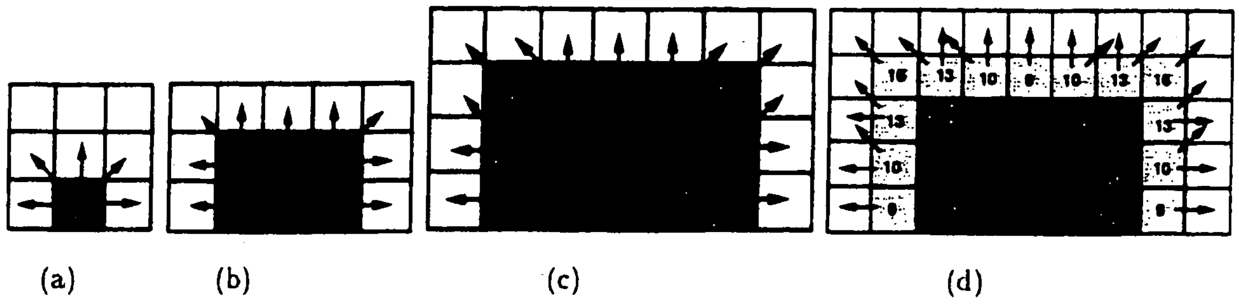


Figure 4.2: This figure illustrates the CEDT algorithm [164] for the upper half of the full picture since the bottom half can be constructed by mirror image. The input object is the black pixel in center. (a) Each pixel on the boundary (just one here) propagates distance values to their immediate neighbors and assign them a vector. Initially there are only three vectors, namely, horizontal, vertical, and diagonal. (b) Here first pixels with value 1 are updated in the horizontal and vertical directions. Then pixels with value 2 are updated in diagonal, horizontal and vertical directions. The horizontal updates assign a horizontal-diagonal direction to the updated pixel and similarly vertical-diagonal direction is assigned to the pixel updated with vertical vector. This completes the initialization stage and there will not be any more changes in the directions. (c) 8 iterations starting with smallest value is shown in one figure for space reasons. (d) Note that some pixels will be updated twice due to the vertical-diagonal or horizontal-diagonal directional masks.

Distance value for a location (x, y) is computed by comparing the distance values of a number of neighbors and choosing the minimum value found. The number of neighboring elements and their weights define the *mask* operators. The size of the mask plays a crucial role in design of distance transform algorithm. The second important factor in the design is in what sequence are these masks are applied, *i.e.*, in how these masks propagate the distance information from the object boundaries. In the ideal distance transform algorithm, each pixel location in the image domain should be updated just the minimal number of times, *i.e.*, *once*. However, this has been a challenging goal in the design of distance transform algorithms.

Thus, distance transform methods may be further classified into two groups based on the propagation of masks in the image domain. These approaches are *raster* scans and *contour-based* scans, Figure 4.1. Raster-scan based Euclidean distance (REDT) transforms [170, 53, 26, 236, 238, 122, 10, 31] have attracted a great deal of interest since they are relatively simpler to implement, and due to independence of the algorithm from embedded shapes. In REDT approach, an image is recursively convolved with a mask with two or more passes in fixed directions. The contour-based Euclidean distance transform (CEDT) approaches has received relatively little attention compared to REDT until recently. In this approaches, the distance information is recursively propagated from the object boundaries towards the inward and the outward regions. Note that this is very similar to the Huygen's construction for the constant speed wavefront propagation. The basic design of CEDT follows ideas originated in Montanari [140], brought to the foreground by more recent work, *e.g.*, Verwer [227], Vincent [229], Ragnemalm [164, 165] and Eggers [59]. While Montanari had the key insight that wave propagation from boundary features was potentially more efficient than raster-scan sequential DT (REDT). Ragnemalm imported the idea of using vectored values for DT [53] and studied different masks and their properties for propagating various metrics from contours. Figure 4.2 illustrates the Euclidean metric propagation on a 2D rectangular distance grid from a point source. Each mask maintains a distance vector (L_x, L_y) from its origin, thus *explicitly* representing the direction and distance of the propagating wavefront. The metric $L^2 = (L_x^2 + L_y^2)$ can also optionally be carried to optimize the number of operations. A further optimization uses *buckets* to store wavefront distance values in order of increasing distance, L^2. Recently, Eggers [59] presented an algorithm which further optimizes Ragnemalm's propagation masks. Vincent [229], motivated by the need for the construction of an error-free distance transform, proposed to encode the object boundaries as chains and to propagate these structures in the image using *rewrite* rules

The simplicity of the REDT has made it popular as compared to the CEDT. However, for simulating wave propagation, CEDT has a key advantage over REDT in that it provides an explicit representation of geometry, wave labels, *etc.* [218, 219]. Other advantages of CEDT include: CEDT is nearly optimal in terms of numerical complexity when compared to raster-scan based DT, and is extendible to 3D by defining additional masks [121]. CEDT is also very efficient in constructing a distance transform or propagating waves in the vicinity of the object boundaries.

While these distance transforms are made computationally efficient for objects with discrete boundary representation, they are not totally error-free. Though distance transforms are proposed which report to be exact [236, 229, 164, 59], in general they are not very intuitive, computationally expensive, rely on the the discrete boundary representation, and some turn out not to be exact. The requirement that the initial sources (edge map) be specified by a discrete boundary, infact, is very limiting since the initial images can provide a richer subpixel specification [55, 202].

4.2 Numerical Solutions of The Eikonal Equation

The numerical solution to the Eikonal equations is not straightforward and special care is required to construct a *stable* and *accurate* numerical scheme. Specifically. the difficulty stems from the computation of the derivatives of the solution. Since the solution may develop singularities (even when the initial boundary is smooth), derivatives are not defined at these points. Thus, the generalized solution must be obtained at those point, *e.g.*, by the viscosity method [128]. We now review three different approaches to construct the solution to the Eikonal equations at these points. Formally, let the front be represented by a 2D curve $\mathcal{C}(s,t) = (x(s,t), y(s,t))$ where x and y are the Cartesian coordinates and t is time. The evolution is governed by

$$\begin{cases} \frac{\partial \mathcal{C}(s,t)}{\partial t} = c(x,y)\vec{N} \\ \mathcal{C}(s,0) = \mathcal{C}_0(s) \end{cases} \tag{4.2}$$

where $\mathcal{C}_0(s) = (x(s,0), y(s,0))$ is the initial curve, and $\vec{N}$ is the unit normal vector. We review numerical approaches to solve this initial value wave propagation problem.

4.2.1 Explicit Front Propagation via Normal Vectors

In this approach the contour is sampled and the evolution of each sample is followed in time by rewriting the Eikonal equation in vector form, namely,

$$\begin{cases} x_t(s,t) = c(x,y)\frac{y_s}{\sqrt{x_s^2+y_s^2}} \\ y_t(s,t) = c(x,y)\frac{x_s}{\sqrt{x_s^2+y_s^2}} \end{cases} \tag{4.3}$$

Specifically, the evolving curve is divided into N equal intervals of size Δs and t is divided into equal intervals of length Δt. Let $X_i^n = (x_i^n, y_i^n) = \{x(i\Delta s, n\Delta t), y(i\Delta s, n\Delta t)\}$ be a numerical approximations of the wavefront. The new front X_i^{n+1} can then be computed from the current front by the following numerical scheme [183]

$$X_i^{n+1} = X_i^n + \Delta t \ \ c(x_i, y_i)\frac{((y_{i+1}^n - y_{i-1}^n), -(x_{i+1}^n - x_{i-1}^n))}{\sqrt{(y_{i+1}^n - y_{i-1}^n)^2 + (x_{i+1}^n - x_{i-1}^n)^2}} \tag{4.4}$$

This evolution is the "Lagrangian" solution since the physical coordinate system moves with the propagating wavefront. Unfortunately, when the normals to the wavefront collide (formation of shocks), this approach exhibits numerical instabilities due to bunching up of sample points, thus requiring special care, such as reparametrization of the wavefront. In addition, topological changes are not handled naturally, *i.e.*, an external procedure is required. This has motivated the curve evolution approach.

4.2.2 The Level Set Method

Osher and Sethian [154] introduced the level set method to track the evolving wavefront in a wide variety of problems. The main idea is to track the evolution of a surface, $\phi(x, y, t)$, whose zero level set always represents the position of the wavefront, $C(s, t) = (x(s, t), y(s, t))$. In this approach, the initial surface, $\phi(x, y, 0)$ is often chosen to be the signed distance function of the initial closed curve, $C_0(s)$; *i.e* $\phi(x, y, 0) = \pm dist(x, y)$ where $dist(x, y)$ is the distance from (x, y) to the initial curve $C_0(s, t)$. Let us now construct the surface evolution whose zero level set propagates according to the Eikonal equation. *i.e.*, $C_t(x, y, t) = c(x, y)\vec{N}$. By differentiating $z = \phi(x, y, t)$ with respect to t at the zero level set, we have

$$\phi_x x_t + \phi_y y_t + \phi_t = 0 \tag{4.5}$$

or

$$(\phi_x, \phi_y)(x_t, y_t) + \phi_t = 0 \tag{4.6}$$

Since $(x_t, y_t) = C_t = c(x, y)\vec{N}$, we have

$$(\phi_x, \phi_y)c(x, y)\vec{N} + \phi_t = 0 \tag{4.7}$$

and the normal vector is given by $\vec{N} = -\frac{\nabla\phi}{|\nabla\phi|}$, since $\nabla\phi$ is always normal to the curve given by $\phi(x, y, t) = 0$. Then, we have

$$(\phi_x, \phi_y)c(x, y)(-\frac{\nabla\phi}{|\nabla\phi|}) + \phi_t = 0 \tag{4.8}$$

$$\phi_t = c(x, y)|\nabla\phi| \tag{4.9}$$

This equation defines only the evolution of points on the zero level set. Similarly, all other points on the surface can be made to evolve according to this equation. This is the Eulerian formulation of the wavefront propagation problem, since the evolution takes place on a fixed coordinate system. The main advantages of this methods are: (i) topological changes are handled automatically, and (ii) geometric measurement from the evolving surface are computed in a robust manner. The numerical solution of this equation, however, still requires special care due to the formation of shocks. Specifically, a stable numerical solution for this evolution is constructed by considering the corresponding *Hamilton-Jacobi* equation, *i.e.*,

$$\phi_t(x, y, t) = c(x, y)H_0(\phi_x, \phi_y, t) \tag{4.10}$$

where $H_0(\phi_x, \phi_y, t) = (\phi_x^2 + \phi_y^2)^{1/2}$. The entropy satisfying numerical algorithms for the Hamilton-Jacobi equations can be found in [154, 155, 193, 120]. The main disadvantage of the level set method is the high computational complexity, due to the additional embedding dimensional, even when the computation is restricted to a narrow band around the curve. Specifically, all points on the embedded surface must be propagated to simulate the evolution of a two dimensional curve.

4.2.3 The Fast Marching Method

We now consider the numerical approximations to the stationary form of the Eikonal equation

$$\nabla u = \frac{1}{c(x,y)} \tag{4.11}$$

where the surface. $u(x,y)$ measures the time at which the wavefront crosses the point (x,y). Rouy and Tourin [171] presented an iterative algorithm for computing a solution to this equation. Specifically, they proposed the following numerical scheme based on the viscosity solutions technique:

$$[max(max(D^{-x}u_{ij},0)-min(D^{+x}u_{ij},0))^2+max(max(D^{-y}u_{ij},0)-min(D^{+y}u_{ij},0))^2] = 1/c_{ij}. \tag{4.12}$$

where

$$\begin{aligned} D^{-x}u_{ij} &= \tfrac{u_{ij}-u_{i-1j}}{\Delta x}, \\ D^{+x}u_{ij} &= \tfrac{u_{i+1j}-u_{ij}}{\Delta x}, \\ D^{-y}u_{ij} &= \tfrac{u_{ij}-u_{ij-1}}{\Delta y}, \\ D^{+y}u_{ij} &= \tfrac{u_{ij+1}-u_{ij}}{\Delta y}. \end{aligned} \tag{4.13}$$

Note that upwinding is used in the computation of derivatives (*upwind* means that information propagates from smaller values of u to larger values.). In this algorithm, the solution at each pixel is iteratively computed until it converges.

Sethian [185, 186] has presented an attractive approach *fast marching* for the solution of the Eikonal equation. In this approach, the solution of the Eikonal equation with an initial value is solved by evolving the wavefront which is represented by a discrete set of pixels. Specifically, the wavefront is propagated ahead in an upwind fashion by solving Equation (4.12) at each grid point on the wavefront. Thus, this method converges to a solution much faster than the one proposed by Rouy and Tourin [171], since the computations are limited to pixels which store the wavefront. In this approach, there are three types of grid points as illustrated in Figure 4.3: (*i*) *alive points* (solid points) are the grid points which the wavefront has already passed and a solution for them has been established; (*ii*) *narrow band*

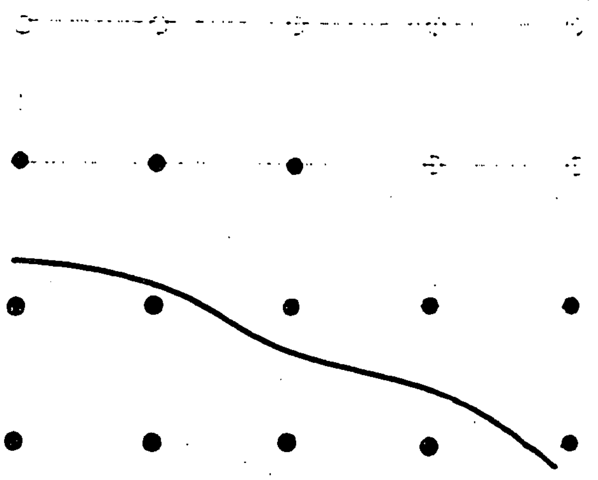


Figure 4.3: This figure illustrates the fast marching algorithm. The analytic wavefront is located between points where the wavefronts has passed (alive points marked as solid), and their immediate neighbors (narrow band points marked as shaded) and the far away points (blank circles) where the wavefront has not reached yet.

points (shaded points) which are the immediate neighbors of the *alive points* and where the wavefront has not reached yet. The analytic wavefront (the curve) may be thought of lying between the alive points and the narrow band points; (*iii*) *far away points* (blank circles) are the set of grid points which do not have any neighboring relations with the alive points. The algorithm consists of an initialization stage followed by a propagation process. In the initialization stage, the given boundary points are marked as alive points, their immediate neighbors are identified as narrow band, and the other grid points are marked as far away points. In each step during the march the followings take place: (*i*) the grid point on the narrow band with a smallest value is removed from the narrow band list and added to the alive points; (*ii*) the four neighbors of this grid point which are either already in the narrow band or if they are in the far away points, they are marked as narrow band points; (*iii*) The values of each neighbor are computed by using Equation (4.12) and selecting the largest possible solution.

This algorithm works well for the solution of the general Eikonal equation, where the front speed is not necessarily constant, is clearly not optimal. Specifically, it does not use any information pertaining to geometry of the front which can be useful in determining direction for propagation. Instead, each grid point on the narrow band visits its four immediate neighbors. However, fewer grid points can be visited if the notion of a direction of propagation is used. In addition, it is not clear why a grid point visits only four neighbors, *i.e.*, diagonal neighbors are not considered as possible update points. Furthermore, the fast marching algorithm does not yield error-free results.

In the following section, we present the discrete wavefront propagation algorithm similar to the fast marching algorithm because the evolving wavefront is represented by a set of discrete pixels. A comparison of our approach to the fast marching will also be given in the following section.

4.3 Discrete Wavefront Propagation

The goal of the wave propagation algorithm is the efficient propagation of the geometric boundary models in the discrete domain and to detect their intersection as a symmetry set. We propose a wave propagation algorithm based on the intermediate view of Huygens' principle and Fermat's principles. In the Huygens' wave propagation construction, each point on the wavefront emanates light rays in the form of a one parameter family of radial curves. This can be applied to discrete domain rather easily, but it is inefficient due to excessive overlap. In Fermat's principle, the light rays propagate orthogonal to the wavefront and the new wavefront may be constructed by moving along the light rays. However, tracing the light rays on the discrete domain would leave gaps on the wavefront. Instead of using a full circle in the first case and a thin beam in the second, we propose using a sector of discrete beams with minimal overlap as in the CEDT algorithm [165], Figure 4.4. Specifically, the CEDT algorithm which typically uses discrete wave directions from discrete boundary locations is now generalized to free-form curve segments, yet maintaining propagation using discrete grid locations and directions. In our approach, instead of propagating only distance information (time), we also propagate geometric models of the sub-pixel boundary. Analytic distance values from the source boundary segments are then computed at each step of propagation process, leading to the proposed Analytic Distance Transform (ADT) algorithm [215]. Appendix A presents the geometric distance functions from the geometric boundary models used in our implementations.

4.3.1 The Discrete Wavefront Propagation Algorithm

In our approach, each piecewise curve segment within a pixel initiates discrete waves in every discrete directions (8) on the discrete grid locations surrounding the boundary segment. Some wave directions are chosen to overlap with each other to cover the whole image

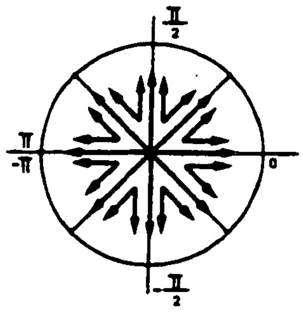


Figure 4.4: This figure illustrates the discrete approximations of the rays. The unit circle is divided into 16 different regions. There are three *main directions*, namely Horizontal (H), Vertical (V) and Diagonal (D). Rays with in these directions are exactly represented on discrete domain. The other regions between those three directions are covered by *approximate directions* namely Horizontal-Diagonal (HD), Vertical-Diagonal (VD) directions which visit two pixels during the propagation.

domain. These waves create a discrete wavefront satisfying a minimum distance value where propagation should continue. The version of the algorithm is described as follows:

(*i*) **Discrete Wavefront Initialization:** Discrete grid points which enclose the sub-pixel boundaries are marked for propagation of discrete waves, Figure 4.5a. These grid points are marked as the *discrete wavefront.*

(*ii*) **Initialization of Discrete Wave Directions:** Each grid location on the *discrete front* can now be considered as a source in analogy to the discrete CEDT. Thus, each source attempts to propagate waves to its immediate neighbors (8 neighbors), Figure 4.5a. When a wave minimizes analytic distance at a point from its boundary source, the *discrete front* will be updated to this grid location. Discrete propagation directions, Vertical (V), Horizontal (H), Diagonal (D) are assigned to each wavefront location based on its location and the orientation of the initial boundary. The two constraints of (*a*) discrete waves should cover the entire propagation space and (*b*) discrete direction overlap should be minimal for efficiency lead to H, V, D and combined HD, VD discrete directions, Figure 4.5b. An additional concern is to prevent duplication of propagation by simultaneously propagating the wavefront. However, since the wavefronts at discrete points arrive at slightly different times, the propagation takes place from the "oldest" wavefront, by ordering distance values at discrete points.

(*iii*) **Propagation From Discrete Wavefront:** The discrete points on the *discrete wavefront* with minimum values are considered as sources for propagation. Among waves arriving

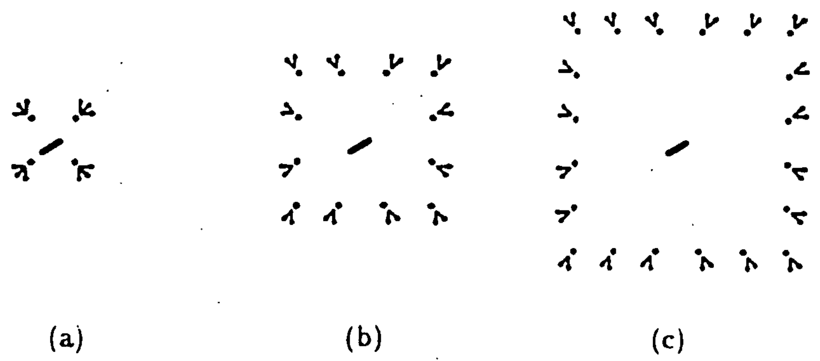


Figure 4.5: The subpixel CEDT algorithm: (a) Geometric description of a subpixel boundary (bold line segment) is propagated to adjacent pixels first. (b) These in turn propagate in discrete directions to their neighbors but resort to analytic computations. (c) This procedure is repeated until all waves have been extinguished or reached the boundary.

at a point the first one (minimal time/distance) is added to the discrete front. This stage is repeated for every point on the *discrete wavefront* which is updated by propagating each point in the direction along which the point had propagated from. Figure 4.5c.

4.3.2 A Comparison With The Fast Marching Method

The fast marching algorithm of Sethian [185, 186] and our discrete wavefront propagation algorithm share two ideas: (*i*) discrete wavefront representation and (*ii*) upwind updates. The fast marching algorithm is designed for the general Eikonal equations, where the front speed is not necessarily constant. However, our approach is specifically designed for the wavefronts propagating with constant speeds. Specifically, first, a discrete grid point will be updated fewer times in our discrete wavefront propagation algorithm since the minimally overlapping discrete directional masks borrowed from Ragnemalm [165] are used to approximate the ray directions. These discrete directional masks are designed such that each grid on a discrete wavefront visits minimum number of pixels, *i.e.* two neighboring grid points at most, whereas fast marching algorithm ignores directional masks and a grid point on a wavefront visits four immediate neighboring points. Second, our approach not only propagates the distance information but also propagates geometric models of the subpixel boundary. This allows us to detect the singularities that form during the wave propagation. Third, the fast marching requires finding the roots of 8 quadratic equations for each grid location in order to solve equation (4.12) numerically. In our approach, just one quadratic equation (for the line and circular arc models) is solved. Indeed in our experiments, we found that our approach is roughly an order of magnitude faster than our implementation

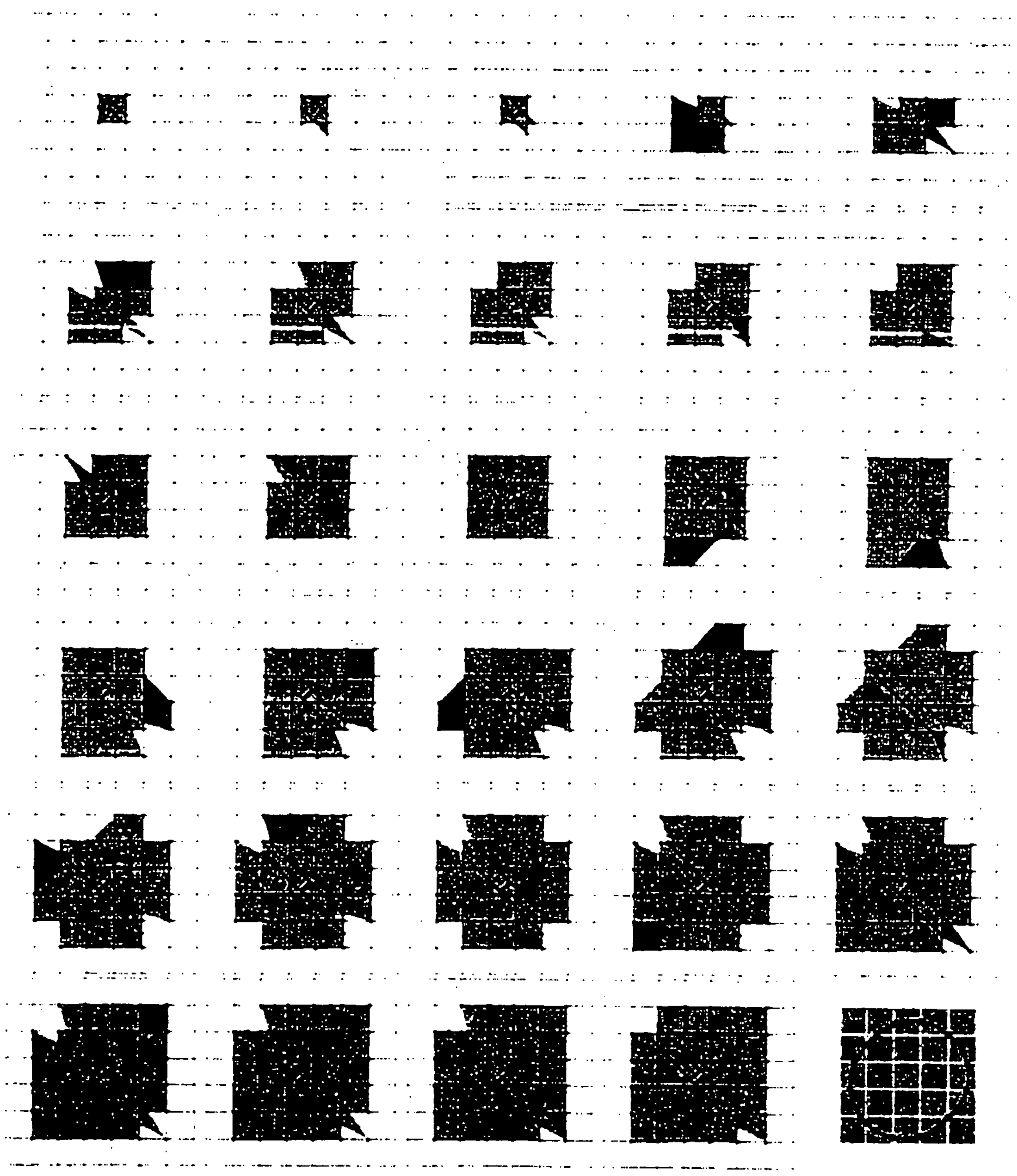


Figure 4.6: This figure illustrates the explicit representation of the *discrete front* for a small subpixel line segment (dark line). In each step, the front is updated either by propagating from discrete grid point sources or from end interval ray sources, thus always maintaining a correct separation between the interior and the exterior of the wavefront. Observe that while the wavefront looks jagged, this is only visual effect due to the fact that each point on the wavefront has arrived at a slightly different, but analytically correct, time f arrival. A subpixel front at any time can be correctly computed as shown in the last box for $t = 2.0$.

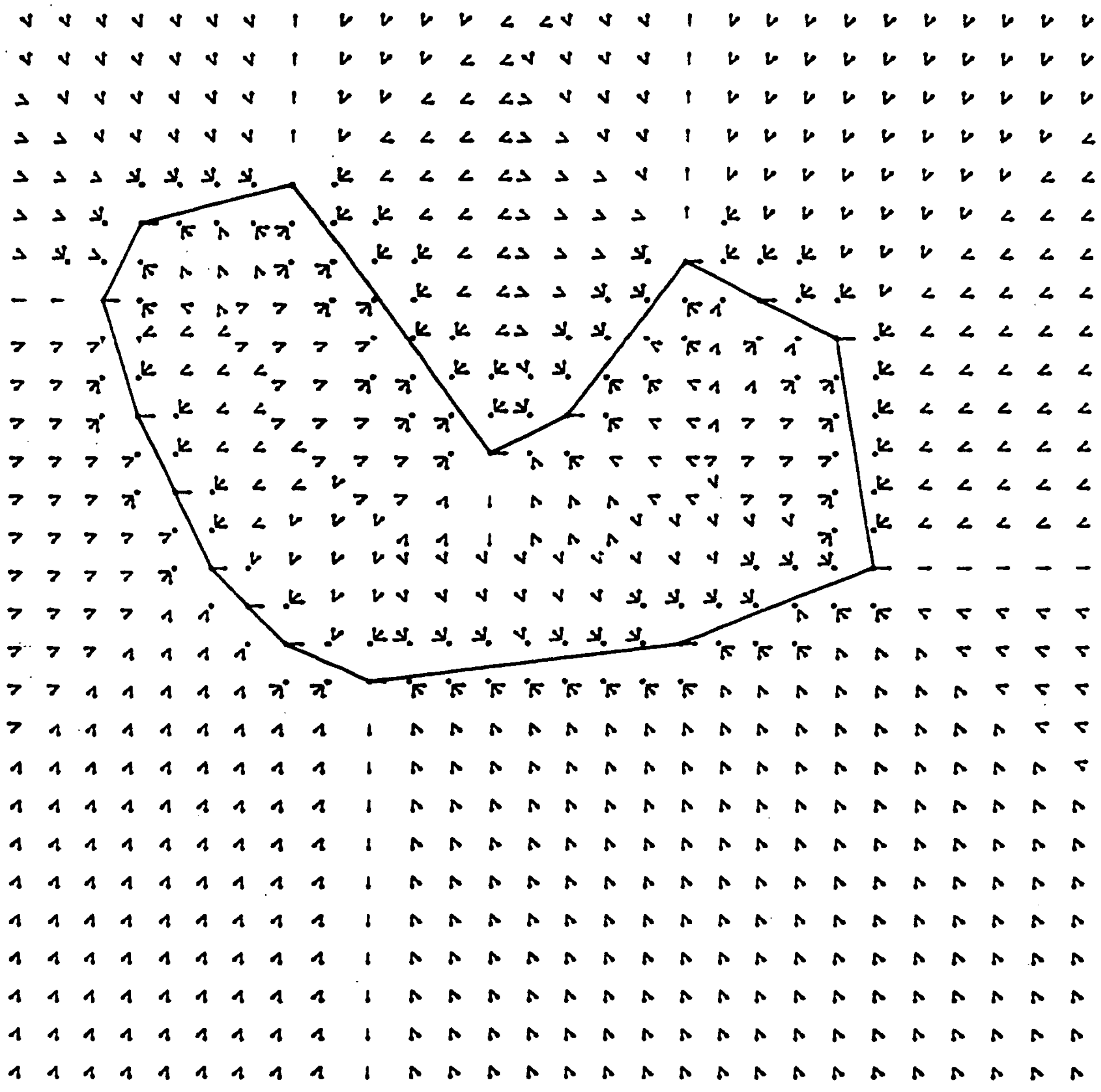


Figure 4.7: This figure illustrates the discrete wave propagation from a shape with subpixel boundaries. The shape boundary is depicted in black. The directional vectors are shown in red.

of fast marching. For a specific example, fast marching took 3.076242 seconds and our discrete wavefront propagation algorithm took 0.380843 seconds for a 200x200 image on a Sparc 10 machine. We are currently investigating the discrete wavefront propagation where the front moves with speed depends on the location of the front, *i.e.*, $c(x, y) \neq constant$.

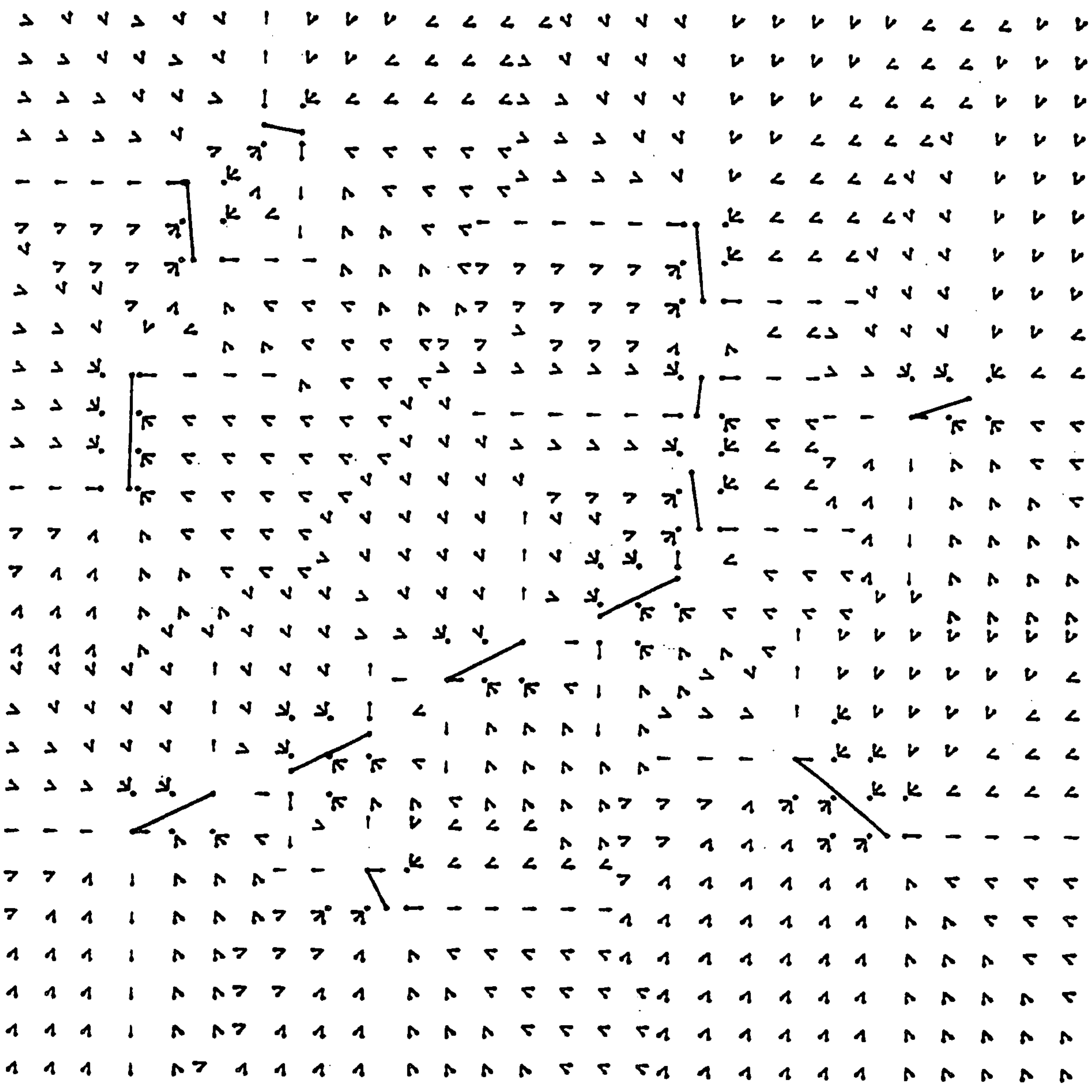


Figure 4.8: This figure illustrates the discrete wave propagation from subpixel line segments. The shape boundary is depicted in black. The directional vectors are shown in red.

4.3.3 Propagation of Errors:

The above wavefront algorithm is almost, but not totally error-free, mainly due to the use of discrete wave directions. The problem stems from the fact that during propagation range of influence of a point [97] becomes very small due to the neighboring waves, Figure 4.9, Figure 4.10. In our algorithm, this influence zone of a boundary model is represented by a set of discrete waves which belong to this model and move on the grid points. Once this

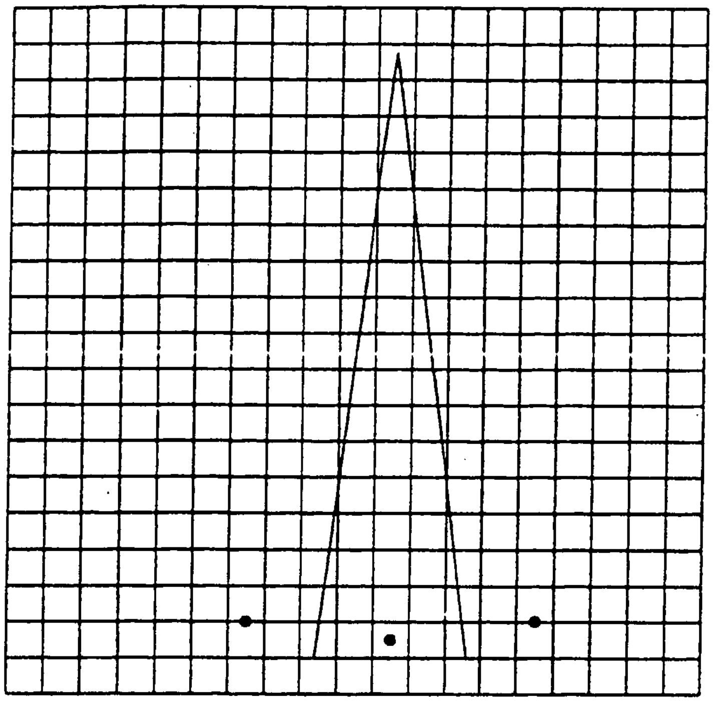


Figure 4.9: This figure illustrates the range of influence of a point with two neighboring points. Note that the range of influence becomes narrower towards the upper portion of the image.

influence zone does not contain any discrete grid location, the discrete waves will not be able to sustain the propagation and they will stop, even though continuous the wavefront continues between pixels. This problem also illustrated in Figure 4.10a with three or more non-connected segments, where middle line's influence zone becomes increasingly narrower and eventually is unable to propagate into the shaded region, due to the waves from left and right boundary intervals, Figure 4.10b. Note that the two grid locations in the shaded area, Figure 4.10b, will receive incorrect waves. It should be emphasized that this problem is independent of boundary representation, *e.g.*, discrete points, lines, arcs.

These errors are not only subject to our discrete wavefront propagation. In fact, discrete distance transforms, upwind numerical methods also produce approximate results in some regions. This problem has already been observed in [229, 165]. These errors could be avoided if the wavefront is explicitly represented on discrete domain which is infact our proposed solution described in the next section. Specifically, in addition to the discrete waves, the boundaries of range of influence of each boundary segments will also propagated for explicit wavefront representation.

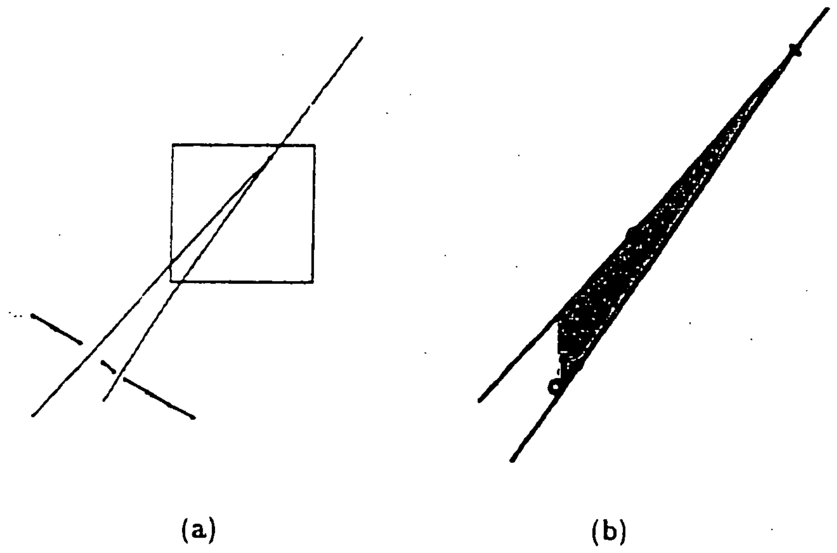


Figure 4.10: This figure illustrates that the discrete wavefront algorithm presented here is almost. but not totally error-free. (a) When the image contains three or more non-connected boundary intervals. The equi-distance lines, the *shocks* of these non-connected boundary segments. marks the end of wave zone for each boundary segment. (b) The problem area which is marked with rectangular window in (a) is shown in detail. Note that waves from the middle line segment will not be able to propagate into the shaded region, but are rather absorbed by the waves from the left and right boundary segment.

Chapter 5

Propagation of Shockwaves

Recall that the main goal of this part is to detect the symmetry map of an edge map of an image. The edge map is represented in the form of curve segments which in degenerate form also include points. These symmetries infact correspond to the singularities in the solution surface of the Eikonal equation [99]. However, the detection of these singularities turns out to be a difficult task. In this chapter, our proposed solution is the *explicit wavefront propagation* which combines the classical curve evolution paradigm and the bisector computation of computational geometry.

5.1 Explicit wavefront propagation

The numerical simulation of curve evolution is governed by stable upwind numerical finite difference schemes [154, 171, 185, 186], which were summarized in Section 4. However, while these methods find solutions in the presence of singularities, our goal is to explicitly recover and represent such singularities to include in the wave propagation process. The detection of such shocks is a rather challenging task. Previously, Siddiqi and Kimia [196] proposed an curve evolution approach to detect singularities on evolving surface and group them using a regularizing shock grammar. The main difficulty of detecting singularities via curve evolution approaches stems from the fact that very accurate curve evolution is an extremely difficult task. Specifically, Figure 5.1 illustrates the *exact* evolution of piecewise curve segments and shows the *exact* singularities on these evolving curves. However, this type of curve evolution is a difficult task since a pixel may contain many curve segments, which obviously cannot be dealt with discrete propagations of upwind numerical finite difference schemes. Thus, an approach for detecting such singularities based on curve evolution must

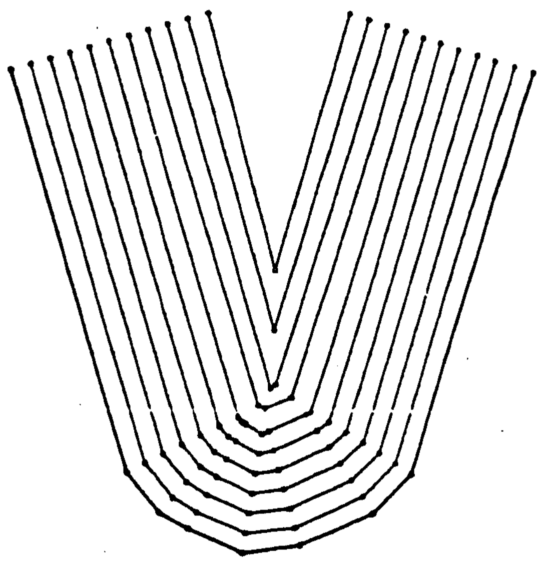


Figure 5.1: This figure illustrates the evolution of a curve and its singularities (shocks). The initial curve is plotted in black, and the evolving curve is red and the shocks (or discontinuities) of the evolving curve is blue. Observe that shocks often fall within a pixel, thus requiring subpixel shock detection. In addition, a pixel may contain more than one curve discontinuity. This requires an *exact* curve evolution and curve models. However, there is currently no such curve evolution algorithm available.

make some approximations, which then result in errors. These errors are often dealt with an additional post-processing, which however may be as difficult as detection process.

When shocks form from the intersection of light rays, they propagate along the curves called *shock paths.* These shock paths along which the discontinuities in solution surface propagate satisfy the integral form of PDE. While in general the determination of a shock path is a difficult task, the shock path for the Eikonal equation, $\nabla u(x, y) = 1$ for a pair of sources with known geometry, corresponds to the equi-distance curves from initial sources. and can thus be analytically computed by finding their *bisector.* This is precisely what is done in computational geometry [16, 166, 167]. Specifically, the bisector of each pair curve segments is computed and the final result is obtained by combining those and pruning non-minimal branches. This is a global and computationally expensive process, and is inconsistent with the local nature of propagation we have adopted. Figure 5.2 illustrates the true (shown in blue) and trimmed (shown in red) bisectors of a line segment and a

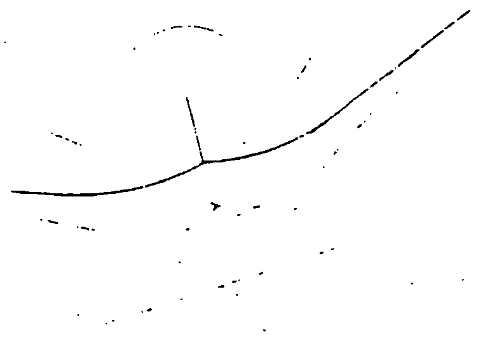


Figure 5.2: This figure illustrates the bisector curve between a circular arc segment and a line segment. The true bisector (blue) is detected by a trimming process.

circular arc segment, where excessive amount of unnecessary bisector computations and then their removal of them take place.

Observe that a true bisector is the subset of a full bisector curve, which has a beginning and an end point. These true bisector curves can be efficiently detected if the beginning and end points of them are correctly identified. In fact, this is the approach that we propose in this paper. Specifically, we propose to detect all initial shock sources, *i.e.*, all points which can give rise to shockwaves, namely second order shocks, A_1^2 and first order shocks arising from curvature extrema, A_3 and propagate these along their analytically known path, the bisector. These shockwaves terminate when intersect with each other, resulting in either a new shockwave or a terminal node, Figure 5.3

Ideally, these initial shock sources can be combinatorially detected for the propagation of shockwaves. However, this can be quite computationally inefficient since only small number of initial shock sources are used to initialize shockwaves. Instead, discrete wavefronts can be used in detection and validation of the correct initial shock sources. Thus, we propose the simultaneous propagation of discrete wavefronts and shockwaves along a mixture of a fixed (Eulerian) grid and a dynamic (Lagrangian) grid. Specifically, on the one hand, discrete wavefronts, which propagate on Eulerian grid, carry full source information so that each shock paths can be analytically determined, but only when necessary, *i.e.*, when two waves collide. On the other hand, shockwaves, which propagate on the Lagrangian grid, carry the sources of colliding wavefronts so that wavefront can be exactly represented during the propagation. This simultaneous propagation results in exact shock detection as well as

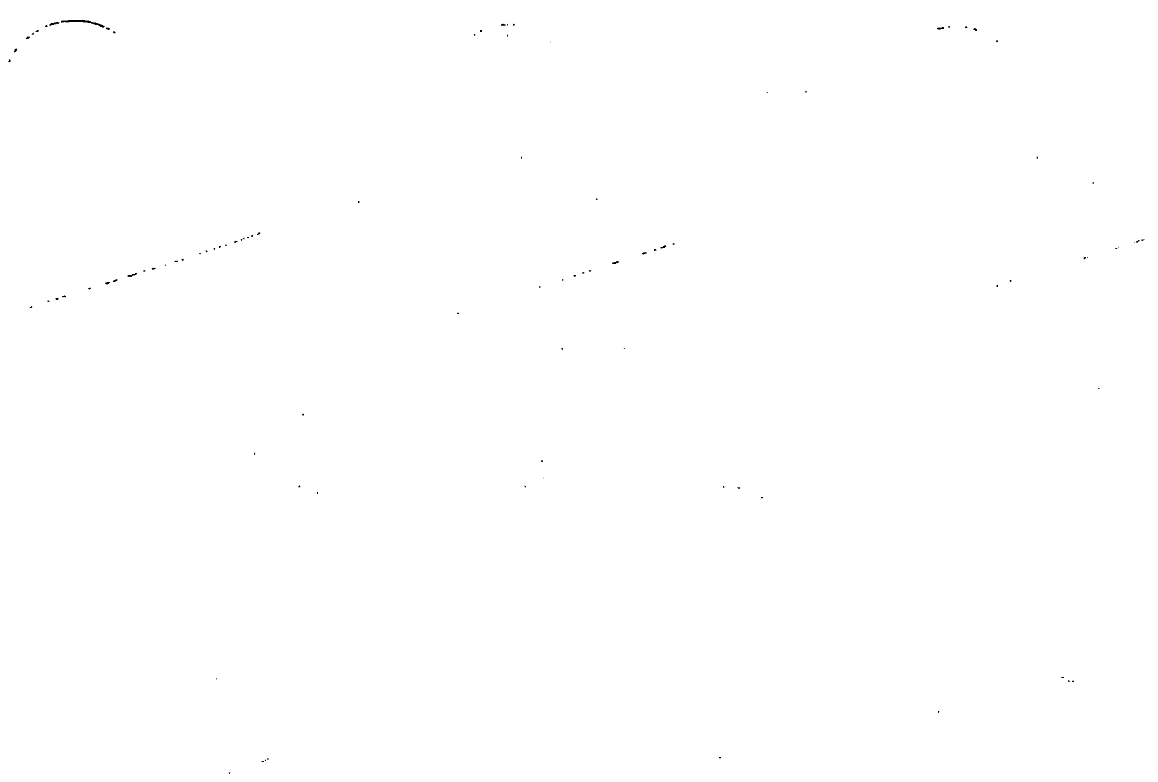


Figure 5.3: This figure illustrates the detection of true bisector curves between a circular arc segment and a line segment by the shockwave propagation algorithm.

exact wavefront propagation on a discrete grid.

5.2 Shock Paths Via Bisectors

We assume that our image edge map consists of N curve segments. each described as a geometric model with a set of parameters $\{P_n\}$ identifying its form and end points. The geometric models include points, line segments, circular arcs, conic arcs, arcs defined by rational functions, and any others for which a distance from the model can be analytically computed. Second, let the distance function from each model be analytically computed as $u_n(x,y)$ where u represents the distance of an arbitrary point (x,y) from curve segment n. Note that $\{P_n\}$ should provide sufficient information for this. Third, we require that the bisector curve between two geometric model can be solved analytically and represented as $B_{nm} = (x_{nm}(s), y_{nm}(s))$, defined as the locus traced by a point (x,y) that is equi-distant from those geometric boundary models n and m , *i.e.*,

$$u_n(x,y) = u_m(x,y). \tag{5.1}$$

Note that the bisector curve for a pair of any combination of point, line segment, and circular arc at most requires solving for a roots of a quartic polynomial which can be done analytically, as described in Appendix B.

5.3 Shockwaves: Sources, Paths, and Sinks

A recent investigation of the local geometry of medial axis points and shocks for closed curves [73] has shown that distinct shock paths can generically only begin at three types of points (i) shock branch end points, A_3, arising from curvature extrema, (ii) second-order shocks, A_1^2, centers of bitangent circles whose radius is locally minimal and grows in both directions along the branch, and (iii) junctions, centers of tritangent circles, where two incoming branches meet with one outgoing shock branch. These are the only generic sources of shock paths, Figure 5.4, for closed curves. For open curve segments, the only additional sources are the end points which give rise to *contact shockwaves* separating regular and rarefaction wavefronts. Second, shock paths themselves consists of curves representing centers of bitangent circles. Third, the shock sinks, points where shock branches terminate, can form either from two incoming shockwaves or from the collision of three incoming shockwaves. These results are important in that they guarantee completeness and correctness of the final results.

Let us first consider an example where two regular curve segments initiate (constant speed) waves simultaneously, Figure 5.5. Specifically, the wavefronts $u_n(x,y)$ and $u_m(x,y)$ originated from boundary sources g_n and g_m, respectively, intersect with each other along the shock path which is given the bisector equation, *i.e.* $B_{nm}(x,y)$. A shockwave, sw_{nm}, which propagates on this shock path inherits information pertaining to the wavefronts, $u_n(x,y)$ and $u_m(x,y)$. This wavefront information is vital in analytically determining the shockwaves forming from the intersection of shockwaves, Section 5.3.4. In addition, the interaction of these two types of waves lead to three types of shocks:

Definition 2 *A shock point arising from two regular waves is regular. A shock point arising from one regular and one degenerate wavefront is* semi-degenerate. *A shock point formed from the interaction of two degenerate wavefronts is* degenerate.

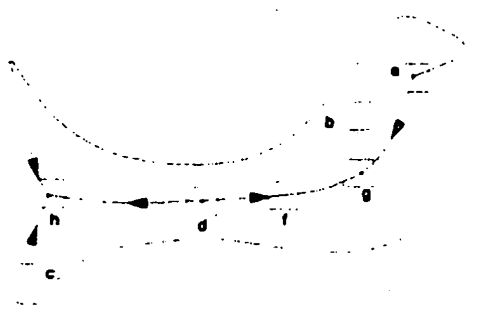


Figure 5.4: This figure sketches the only generic possibilities for shock formation and propagation superimposed on a rectangle between four grid crossings. (a) original shape. (b) no wave collision. (c) formation of first-order shock, (d) a second-order shock. (e) a junction with an outgoing branch. (f) propagation of a shockwave, (g) termination of two branches at a fourth order shock. (h) termination of three branches at a fourth order shock.

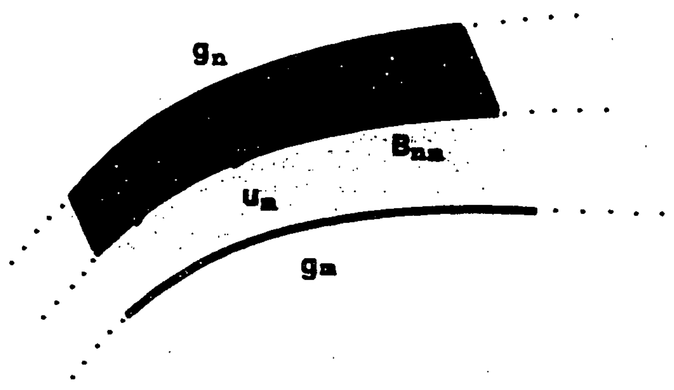


Figure 5.5: This figure illustrates the shock path of two wavefronts. $u_n(x, y)$ and $u_m(x, y)$ originated from the boundary sources g_n and g_m respectively.

In this paper, green, blue and red colored shocks correspond to the regular. semi-degenerate. and degenerate shocks, respectively.

5.3.1 Shockwaves from boundary extrema

The light rays originating from a regular curve intersect with each other *first* at a cusped point which corresponds to the center of a curvature extrema of the curve, Figure 5.6a. Note that not every curvature maxima on a curve segment is able to initiate a shockwave because it is possible that the center of curvature maxima of a boundary model lies outside the influence zone of the boundary model, Figure 5.6b. For examples, points line segments and circular arcs do not have curvature extrema, *e.g.*, in contrast to parabolic segments.

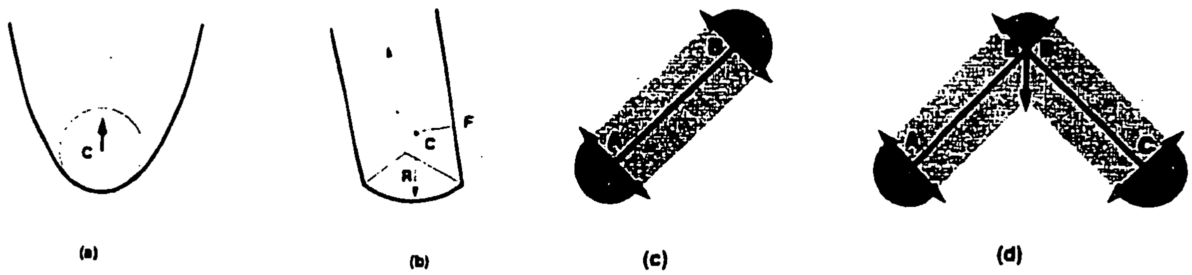


Figure 5.6: This figure illustrates the formation of shockwaves. Specifically, (a) a shockwave forms from a center of a curvature maxima, (b) No shockwave will be initialized from the center of a curvature maxima of a boundary model if this point, C' lies outside the influence zone of the boundary model. (d) End points of a curve segment initiate contact shockwaves as representing the front between regular waves (light shaded regions) and rarefaction waves (dark shaded regions). (e) A discontinuity initiates one regular shockwave and two contact shockwaves.

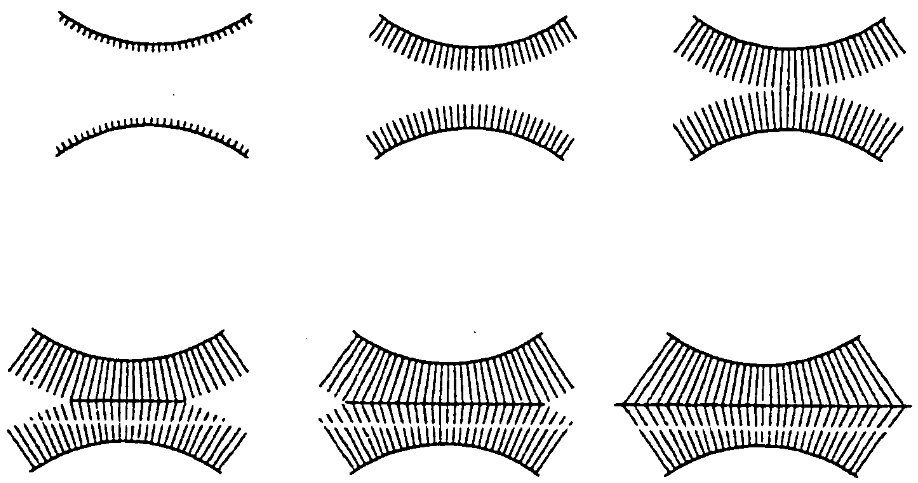
5.3.2 Contact shockwaves from the end points of boundary segments

Each geometric boundary model initiates two regular waves from the boundary segments with true orientation information, and two rarefaction waves from the end points of the boundary model. The rays that separate rarefaction and regular waves are ***contact shockwaves***. Thus, each geometric boundary model initiates two contact shockwaves from each end point, thus four in total, Figure 5.6c. Sometimes, it is possible that two boundary segment create a discontinuity due to overlapping end points. For example, Figure 5.6d illustrates a case where the boundary segments AB and BD have the same end points. In this case, two of the four contact shockwaves initiated collide with each other instantaneously and terminate, and form a junction which then leads to a regular shockwave.

5.3.3 Shockwaves from second-order shocks

A second-order shock forms when two rays from two boundary segments intersect with each other at a point, Figure 5.7a, breaking each wavefronts into two piece, Figure 5.7b. A second-order shock is a source for two shockwaves in opposite directions which are orthogonal to the rays that form the second-order shock. The timely and accurate detection of second-order shocks is a crucial factor in determining the shocks of unorganized curve segments since shockwaves and wavefronts propagate simultaneously. In keeping with the local nature of this approach, second-order shocks must not be detected a-priori, but at the time of formation, to avoid unnecessary and expensive computations. Specifically, in each step of

(a)



(b)

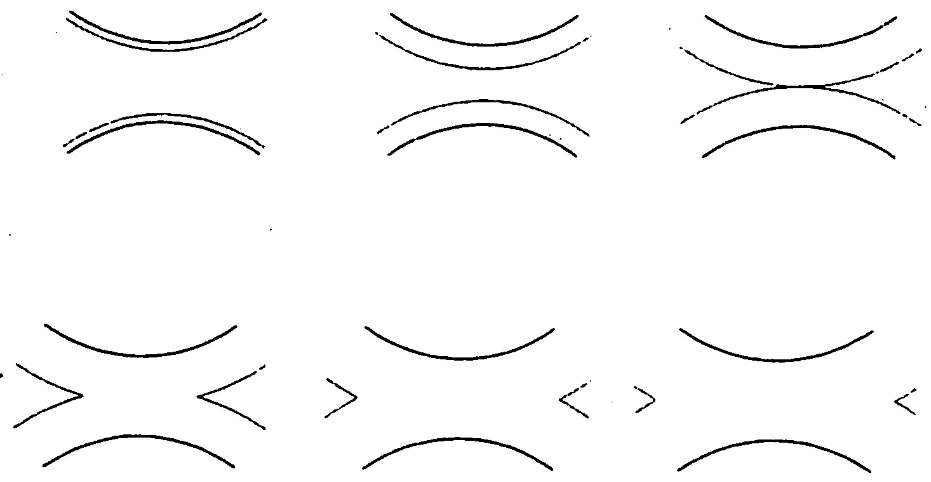


Figure 5.7: This figure illustrates formation of a second-order shock from two geometric boundary models. When two parallel but opposite direction rays collide with each other at a place a second-order shock forms. This second-order shock initialize two shockwaves in opposite directions. (a) propagation of rays and formation of the second-order shock. (b) propagation of the wavefront.

the (shockwave or wavefront) propagation, each updated grid location inherits a source and its geometric model parameters through the discrete wavefronts. When a grid location has received two (or more) distinct waves, a second-order shock is analytically computed for the pair of two geometric boundary models k and l coincident there. Let the distance functions corresponding to sources k and l be represented by $u_k(x,y)$ and $u_l(x,y)$, respectively. The

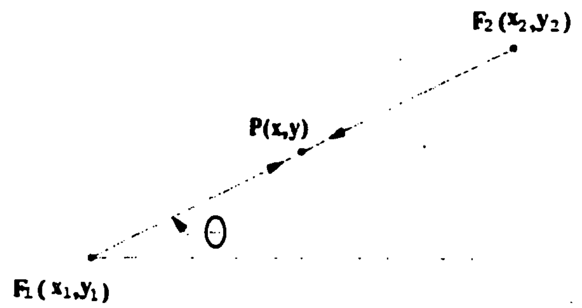


Figure 5.8: A second-order shock at $P(x,y)$ forms from the rays originated from the boundary points $F_1(x_1,y_1)$ and $F_2(x_2,y_2)$.

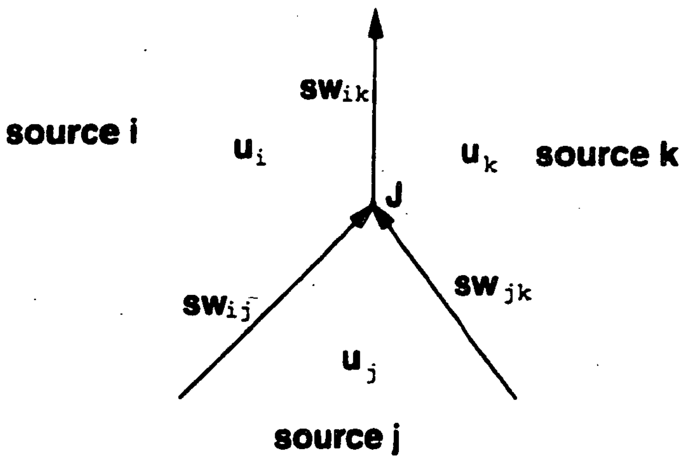


Figure 5.9: The collision of the shockwaves sw_{ij} and sw_{jk} at the junction point, J, triggers a new shockwave, sw_{ik} which propagates on a shock path that separates u_i and u_k.

second-order shock is obtained by solving

$$\begin{cases} u_k(x,y) = u_l(x,y) \\ \vec{N_k}(x,y) = -\vec{N_l}(x,y) \end{cases} \tag{5.2}$$

Specifically, we will use the following proposition to compute the second-order shocks for these geometric models, *point, line, circular arc.*

Proposition 1 *Let a second-order shock at $P(x,y)$ form from two boundary sources. Then the location of the second-order shock is then given by*

$$\begin{cases} x = \frac{x_1+x_2}{2} \\ y = \frac{y_1+y_2}{2} \end{cases} \tag{5.3}$$

where $F_1(x_1,y_1)$ and $F_2(x_2,y_2)$, are the foot points of $P(x,y)$ on the boundary source 1 and 2, respectively. Also the direction of the two wavefronts it gives rise is orthogonal to F_1F_2.

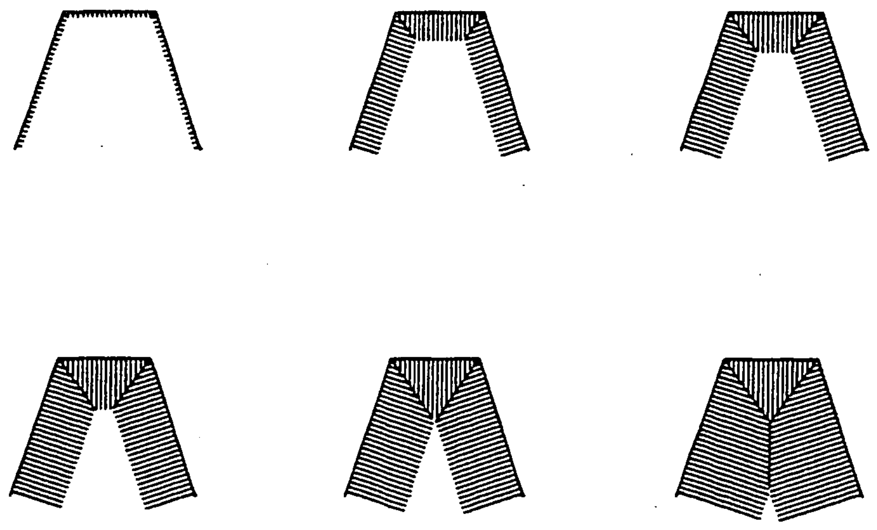


Figure 5.10: This figure illustrates the formation of a shockwave from a junction point. Specifically, two shockwave (red) intersect at a junction point, a new shockwave forms from this point.

Proof: By definition, a second-order shock is equi-distant from its foot points, $F_1(x_1, y_1)$ and $F_2(x_2, y_2)$, *i.e.*, $|PF_1| = |PF_2| = \frac{|F_1F_2|}{2}$, Figure 5.8. Equation 5.3 follows.

The second-order shock computation for a number of geometric boundary models, namely, *point*, *line*, and *circular arc* is presented in Appendix C.

5.3.4 Shockwaves from junctions

When two shockwaves collide, a new shockwave forms from the collision point, or *junction* (these are A_1^3 points). Specifically, suppose that two shockwaves $sw_{ij}(x, y)$ and $sw_{jk}(x, y)$ collide at a junction point, $J = (x_j, y_j)$ where both shockwaves terminate, Figure 5.9. The behavior of u beyond this point is determined by a new shockwave from the wavefronts, u_i and u_k respectively. Observe that the accurate computation of junction points plays an important role in the approach presented in this paper, since they are the source of new distinct shockwaves. The junction point can be analytically computed from equating two bisectors, namely

$$B_{ij}(x, y) = B_{jk}(x, y) \tag{5.4}$$

For the space of geometric models up to circular arcs, the junction computation at most requires solving the roots of a quartic polynomial which can be solved analytically. Table 5.1 summarizes maximum degree of the polynomial for these geometric boundary models. If higher order geometric models are used for boundary representation numerical approximations may be required for junction detection.

degree of polynomial / junction type	linear	quadratic	quartic
point/point/point	x		
point/point/line		x	
point/point/circle		x	
point/line/line		x	
point/line/circle			x
point/circle/circle			x
line/line/line	x		
line/line/circle		x	
line/circle/circle			x
circle/circle/circle			x

Table 5.1: This table summarizes the degree of the polynomial of a junction from different geometric boundary sources, namely point, line, circular arc.

Chapter 6

Discrete Wavefront and Shockwave Propagation: The algorithm

In this chapter. we first present the data structures that are used in our algorithm. followed by an explanation of the algorithm using a flow diagram and a concrete example. Second. we examine the computational complexity of our approach. Third. we present several illustrative examples.

6.1 Overview Of The Algorithm

The wave propagation algorithm is based on the *integrated* and *simultaneous* propagation of discrete wavefronts on a discrete grid, on the one hand, and shockwaves on an analytically determined path, on the other. In the previous chapter. we described the discrete wavefront propagation algorithm and concluded that this algorithm does not yield error-free results when waves continue between pixels without the discrete grid representing them. Figure 6.1. The remedy we proposed in In Chapter 5 is the propagation of shockwaves enables us to represent the wavefront explicitly. This algorithm then has two propagation aspects. the discrete grid wavefront propagation and the shock propagation. In addition. our algorithm is divided into two main stages: initialization and propagation. The flow diagram of the detailed algorithm is shown in Figure 6.2. In addition, Figure 6.1 illustrates the propagation of discrete waves and shockwaves on an example in detail.

In the first stage, the initialization stage, discrete wavefronts are initialized from all geometric boundary models, by marking all the discrete grid points neighboring all geometric boundary models as the discrete wavefront, Figure6.1b, Figure 6.2, Module 1. Each point on a discrete wavefront stores: (*i*) the closest geometric boundary model as the source of the wavefront at that point and as an aid to efficiency, (*ii*) the distance to this geometric boundary model, and (*iii*) the discrete directions approximating the exact propagation direction of the wavefront, Table 6.1. If a discrete grid point is a neighbor to more than one geometric boundary model, a second-order shock is computed from each of these geometric boundary models, Figure 6.1b, Figure 6.2, Module 3.

Second, the remaining initial shockwaves, namely, those from the centers of curvature maxima of geometric boundary models and the contact shockwaves from end points of the geometric boundary models are initialized, Figure 6.2, Module 2. Together with second-order shocks, these are the *only* sources for initializing shockwaves [75]. These discrete wavefronts and shockwaves are then stored in the *ordered wavefront list* which is sorted according to increasing distance values (time of formation).

In the second stage, the discrete wavefronts and shockwaves are then recursively propagated by choosing and propagating the wave with smallest distance from the ordered wavefront list until all the waves either terminate or leave the image boundaries, Figure 6.2, Module 4. First, discrete wavefronts are propagated as follows, Figure 6.2, Module 5: a discrete wavefront propagates to its neighboring discrete grid points indicated by its discrete directions. The propagation of a discrete wavefront to a discrete grid point requires the computation of the distance from this point to the source of the visiting wavefront and comparing it to the existing distance value(s) at this discrete grid point. If the new distance is smaller, this discrete grid point is marked as a discrete wavefront and is inserted in the ordered wavefront list. In addition, the discrete directions, which specify where this discrete wavefront must visit in the next iterations, are inherited from the discrete wavefront that has marked this discrete grid point as the discrete wavefront. Whenever a discrete wavefront visits a discrete grid point, the discrete wavefront is stored in the *quenched wavefront list* of this discrete grid point, regardless of the distance value that it brings to this point. If there are more than one distinct wavefronts in this list a second-order shock is computed, Figure 6.2, Module 3.

Second, shockwaves are propagated as follows, Figure 6.2, Module 6 and 7: When a shockwave has just been initialized, its shock path is computed and stored, Figure 6.2, Module 6. If the shockwave is a fourth-order shock, it is marked as a terminal node and it is removed from the ordered wavefront list. An initialized shockwave then propagates on the intersection of grid lines with its shock path, Figure 6.2, Module 7, until it reaches the end point of its shock path, Figures 6.7 and 6.1. When a shockwave flows to its end point, where it intersects with another shockwave, it is terminated and a new shockwave forms from this point which is then inserted in the ordered wavefront list for propagation. In analogy to the discrete wavefront propagation, a shockwave stores its source information to the quenched wavefront list of the discrete grid points that it visits and a second-order shock is computed if there are more than one wavefronts in the quenched wavefront list of these discrete grid points. Let us now present each module in this diagram in detail:

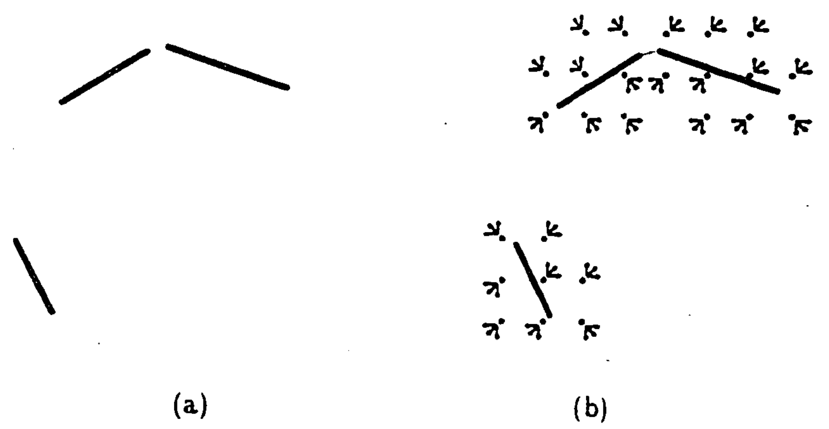
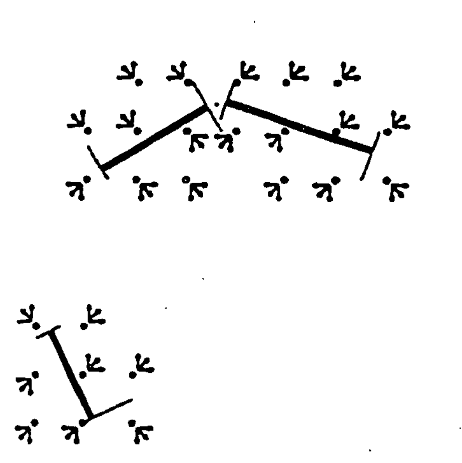
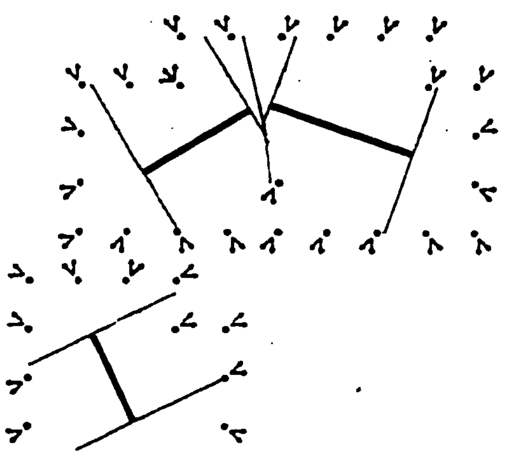
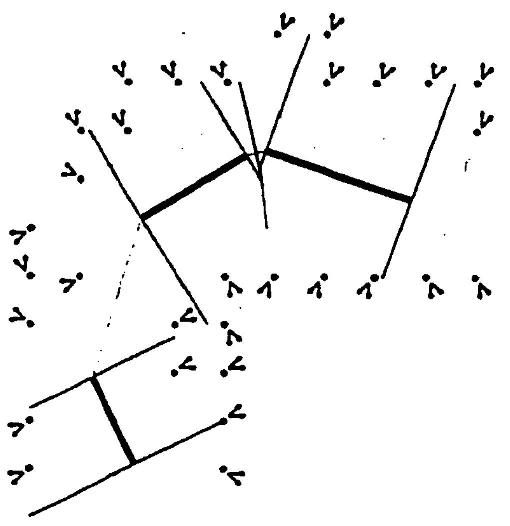


Figure 6.1: (a-b): This figure illustrates the simultaneous propagation of *DiscreteWaveFronts* and *ShockWaves*: (a) Input image; (b) *DiscreteWaveFronts* are initialized on the neighboring *DiscreteGridPoints* that enclose the *GeometricBoundaryModels*. These *DiscreteWaveFront* points are marked as initial and visit at most three *DiscreteGridPoints* to completely cover the image domain. Note also that a second-order shock forms between the two top edgelets, or *GeometricBoundaryModels* and a blue line is drawn between the foot points of this shock to illustrate it.


(c)


(d)


(e)

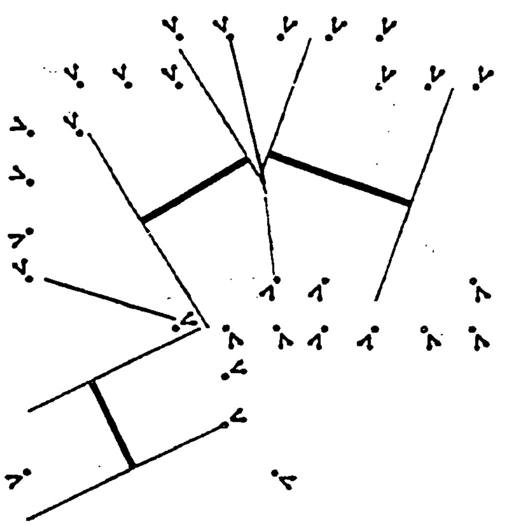
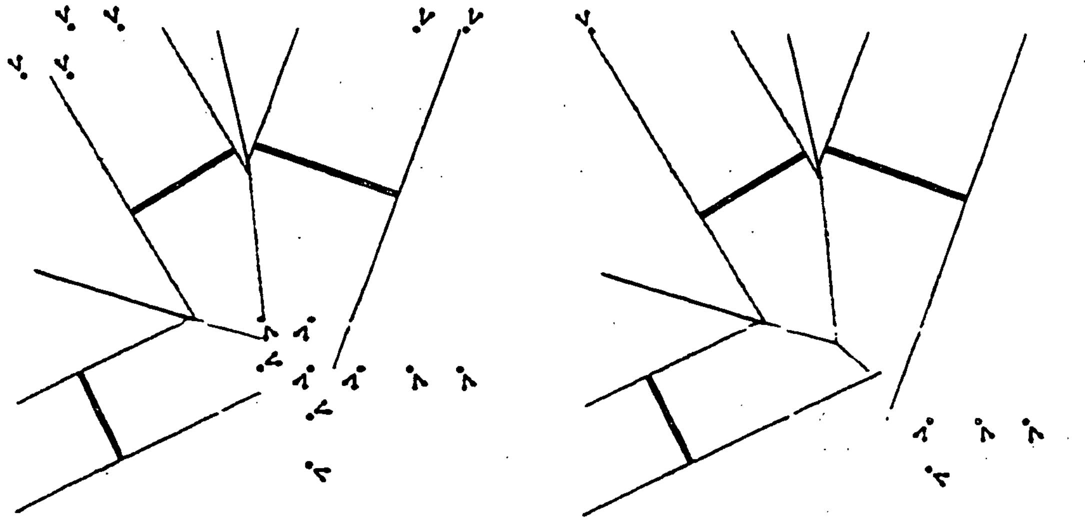

(f)

Figure 6.1: (c-f): (c) Propagation of contact *ShockWaves* (grey) to *next grid crossing*, which is the intersection of its *ShockPath* with the next grid line. (d) The initial stage of *DiscreteWaveFront* propagation is done. From this stage *DiscreteWaveFronts* visit at most two *DiscreteGridPoints*. Observe how the second order shock gives rise to degenerate shock branches; one branch continues as a degenerate branch, while other collides with a contact *ShockWave* and becomes a semi-degenerate (blue). After collision with another contact *ShockWave*, this becomes a regular *ShockWave* (green). (e) Another second order shock has formed, as illustrated by the light blue line between the two left most *GeometricBoundaryModels*. (f) Two degenerate *ShockWaves* are propagating along with the *DiscreteWaveFront*.


(g) (h)

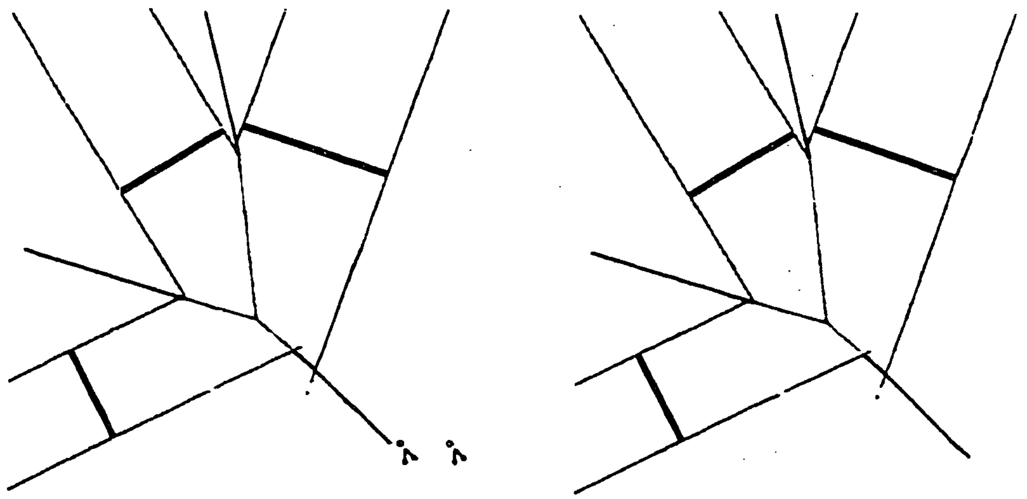

(i) (j)

Figure 6.1: (g-j): (g) One of these *ShockWaves* collides first with one contact *ShockWave* and then another to become a regular *ShockWave* (green). (h) Two regular *ShockWaves* collide and launch a new *ShockWave* (green). (i) This regular *ShockWave* collides with one contact *ShockWave* and becomes a semi-degenerate branch (blue). It then collides with another contact *ShockWave* and becomes a degenerate branch. (j) All propagation has stopped. The resulting *ShockWaves* are already organized as a graph. This algorithm returns this graph as well as the error-free distance transform.

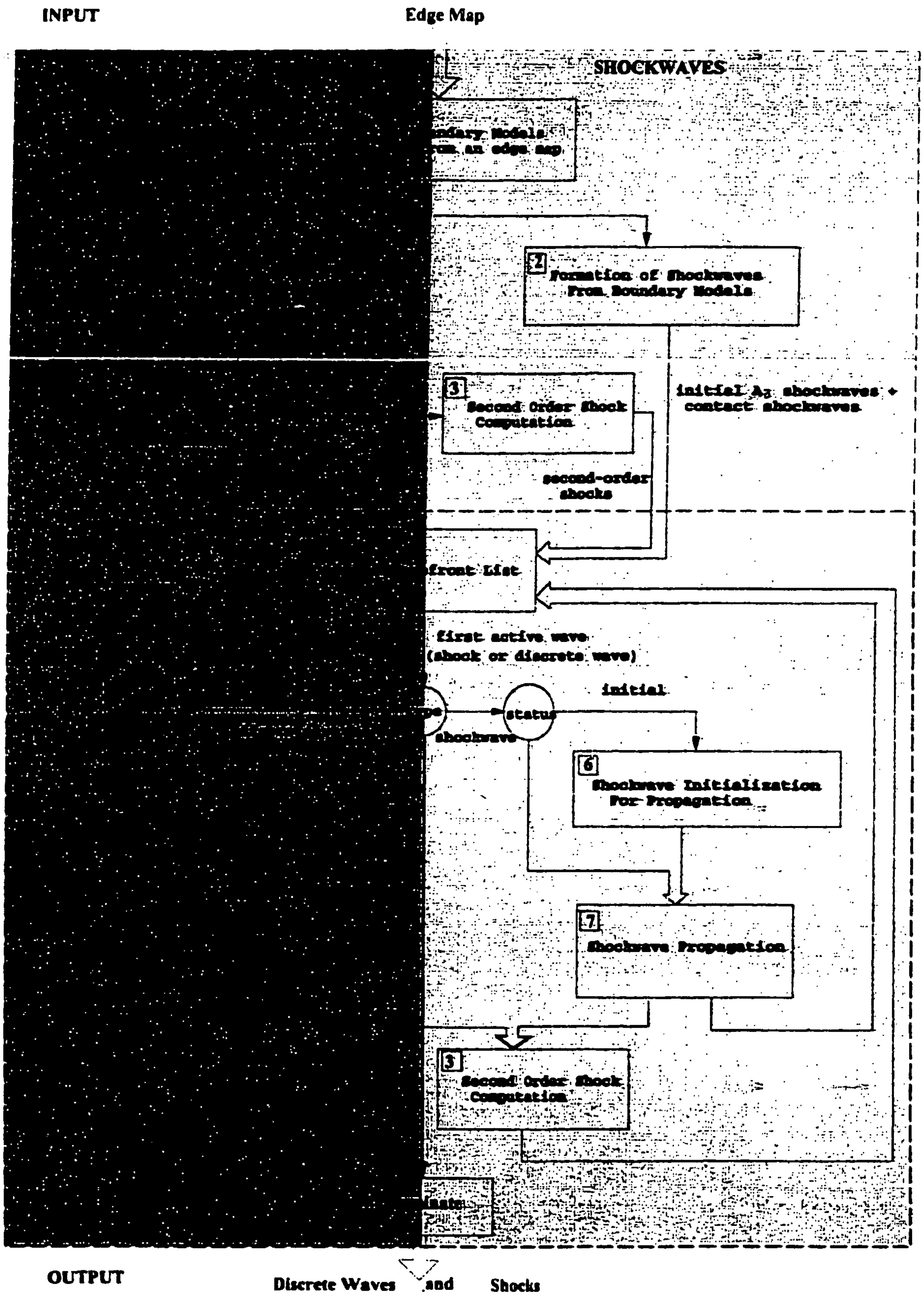

INPUT
Edge Map
SHOCKWAVES
ndary Models
rom an edge map
2 Formation of Shockwaves From Boundary Models
initial A3 shockwaves + contact shockwaves
3 Second Order Shock Computation
second-order shocks
front List
first active wave (shock or discrete wave)
initial
status
shockwave
6 Shockwave Initialization For Propagation
7 Shockwave Propagation
3 Second Order Shock Computation
OUTPUT
Discrete Waves and Shocks

Figure 6.2: The flow diagram of the shock detection algorithm.

6.2 Data Structures

Discrete Grid Point (*DiscreteGridPoint*): is a point (x, y), where x and y take on integer values and which stores

- the distance value from the closest *GeometricBoundaryModel*. This is initially "null", but is updated until convergence.
- the type of *WaveFront*. *i.e.*, regular or degenerate, from the closest source that has visited this point.
- a list *QuenchedWaveFrontList*, a substructure of *DiscreteGridPoint*. that holds the *WaveFronts* that have visited this grid point. This list is initialized to "null".

Discrete Grid (*DiscreteGrid*): consists of

- MxM discrete grid points (*DiscreteGridPoints*).
- M vertical and M horizontal lines

Geometric Boundary Model (*GeometricBoundaryModel*): Each boundary model is the source of a wavefront. The *WaveFront* propagates the set of parameters which completely identify this *GeometricBoundaryModel* until it annihilates due to collision with other wavefronts. The parameters identifying each boundary model. or curve segment. are

- location of the curve segment end points (in relative coordinates if implemented in parallel), and the parameters sufficient to completely describe the shape of the *GeometricBoundaryModel*. For example, for a line we use (x_0, y_0, x_1, y_1) where (x_0, y_0) and (x_1, y_1) are the end points. while for a circular arc, we use $(x_0, y_0, x_1, y_1, x_c, y_c, r)$ where (x_0, y_0) and (x_1, y_1) identify the two end points. (x_c, y_c) is the center of the circle, and r is the radius of the circle.
- *ShockWaves* that are initiated from it.
- a list *UsedGeometricBoundaryModelList*, a substructure of *GeometricBoundaryModel*, which stores other *GeometricBoundaryModels* which have quenched with it. Since the collision of *WaveFronts* from two *GeometricBoundaryModels* might be signaled in the discrete domain several times. this list prevents duplicate second-order shock computations. This is not necessary if efficient is not a concern of if the implementation is using parallel architecture.

Wavefront (*WaveFront*): The wavefront itself is a closed curve that separates the region where the initiated disturbance from the boundary model has passed and the region that it has not reached yet. A discrete wavefront surface, $u(x,y)$ is an implicit representation of this curve where u measures the distance of (x,y) from the source, *i.e.*, $u(x,y)$ is the time of arrival of the *WaveFront* at the point (x,y). When a portion of the *WaveFront* collides with another, it terminates at a collision point (discrete grid point) and initiates a *ShockWave*. These *WaveFronts* are said to be neighbors after the collision and each marked as such. Thus, each *WaveFront* carries at each discrete point (x,y) (with integer coordinates):

- time of arrival value $u(x,y)$, *i.e.*, distance from a point (x,y) to the source model.
- source, *GeometricBoundaryModel*, *i.e.*, the parameters specifying the source boundary model.
- neighboring *WaveFront*: These will be determined by the *ShockWaves* which form from their collision.
- direction of the *WaveFront* with respect to the normal direction of its *GeometricBoundaryModel* source. This is one if the *WaveFront* propagates in the normal direction specified by the *GeometricBoundaryModel*, otherwise it is *Null*.

Discrete Wavefront (*DiscreteWaveFront*): The discrete wavefront is a set of *DiscreteGridPoints* which explicitly represent the frontier of the *WaveFront* propagated on the discrete grid *at a specific time*. These are the active, yet to be propagated points. Each *DiscreteGridPoint* (x,y) on a *DiscreteWaveFront* stores:

- source of the *WaveFront*, *i.e.*, the parameters specifying the *GeometricBoundaryModel*;

- discrete directions, in which the discrete wavefront should propagate; and
- distance value which is measured from the source of the *WaveFront*.

Shockwave (*ShockWave*): Each shockwave propagates on an analytically determined path that separates two colliding *WaveFronts*. Each *ShockWave* carries:

- the *WaveFronts* that collide to form it.
- parameters that define the ***shock path***: These are determined from the two colliding *GeometricBoundaryModels*. For example, the shock path of a *ShockWave* formed from a point and a line source is a parabola specified by $ax^2+by^2+cxy+dx+ey+f=0$, thus the set of parameters (a,b,c,d,e,f) is sufficient to determine the shock path. Intrinsic representations are, however, preferable.
- distinct points of the shock path.

  (i) initial point, $I(x,y)$ where a *ShockWave* initially forms, (*ii*) current sample point, $C(x, y)$ which specifies the location of a shockwave on the *ShockPath*; these sample points are limited to be on the intersection of a shock path and a grid line;

(*iii*) end point, $E(x, y)$ where a *ShockWave* is expected to intersect with another neighboring *ShockWaves*; note that the end point is constantly changing as new shockwaves update the scene.

- ancestor and descendent *ShockWaves*.
- distance from the source of each *WaveFront* giving rise to the *ShockWave*.
- neighboring (left and right) *ShockWaves* which are determined from the colliding *WaveFronts*.
- order of shock, *i.e.* first-, second-, third-, fourth-, or contact.
- shock label, *i.e.*, regular, semi-degenerate, degenerate.

Ordered Wavefront List (*OrderedWaveFrontList*): This is a global queue which stores the distance from source for discrete wavefronts and shockwaves in an ordered manner. This is only needed for a sequential machine to ensure "simultaneous" propagation, solely for efficiency.

6.3 Modules

1. Initialization of Discrete Waves from Boundary Models: *DiscreteGridPoints* which enclose the *GeometricBoundaryModels* are marked for the propagation of *Discrete WaveFronts*, Figure 6.1b. These grid points are also used to construct the *Discrete Wave Front*. Each *DiscreteGridPoint* on the *DiscreteWaveFront* is assigned (*i*) an analytic distance value, (*ii*) source of each *WaveFront*, *i.e.*, the *GeometricBoundaryModel*, (*iii*) initial discrete directions which specify the *DiscreteGridPoints* that this *WaveFront* must visit in the next iteration, and, (*iv*) a flag which indicates whether this is an initial propagation. The initial discrete directions of each grid point are determined from the discrete approximations of the ray directions specified in this grid point. Table 6.1 summarizes these directions for a *DiscreteGridPoint*. Observe that in the initial stage, each *DiscreteGridPoint* on the *WaveFront* must visit the three immediate neighbors, *i.e.*, horizontal, vertical, and diagonal neighbors if the ray direction is not a purely horizontal or vertical ray. This is required in order to cover the entire image domain.

In addition, each *DiscreteGridPoint* stores the *WaveFronts* that visit it. Specifically, when a *DiscreteWaveFront* visits a *DiscreteGridPoint*, its source information is stored in

update direction from (x,y) / ray directions	update direction 1	update direction 2	update direction 3
direction = 0	(x+1,y)		
0 < direction < 90	(x+1,y)	(x,y+1)	(x+1,y+1)
direction = 90	(x,y+1)		
90 < direction < 180	(x-1,y)	(x,y+1)	(x-1,y+1)
direction = 180	(x-1,y)		
180 < direction < 270	(x-1,y)	(x,y-1)	(x-1,y-1)
direction = 270	(x,y-1)		
270 < direction < 360	(x+1,y)	(x,y-1)	(x+1,y-1)

Table 6.1: This table specifies the *DiscreteGridPoints* where an **initial** *DiscreteWaveFront* at a grid point (x, y) visits in the next iteration.

the *QuenchedWaveFrontList* of this grid. When a *DiscreteWaveFront* visits a grid point and there are other *WaveFronts* in the *QuenchedWaveFrontList* at that point a quench is signaled which launches the computation of a second-order shock.

2. Formation of Shockwaves from Boundary Models: Each *GeometricBoundaryModel* initiates two contact *ShockWaves* moving in opposite directions from each of its end points and a *ShockWave* from the centers of curvature maxima of the *GeometricBoundaryModel*, if any. These *ShockWaves* with their associated time of formation are inserted in the *OrderedWaveFrontList* in order of increasing time. A *ShockWave* from the curvature maxima is marked as initial, but it is not a-priori known whether this *ShockWave* ever forms (Recall that if the center of a curvature maxima falls outside the influence zone of the *GeometricBoundaryModel* no *ShockWave* is initialized from this point).

3. Second-Order Shock Computation: When a *WaveFront* quenches with the other *WaveFronts* on a grid point, a second-order shock is computed from the recently arrived *WaveFront* source, and the other *GeometricBoundaryModels* stored there previously. Observe that often two *WaveFronts* collide with each other in more than one *DiscreteGridPoints*, leading to duplicating this computation. To eliminate unnecessary second-order shock computations, this computation is recorded for each *GeometricBoundaryModel*. Alternatively, this record could be maintained at each pixel's four corner grid points, thus maintaining the local nature of the wave propagation algorithm. When a second-order shock is successfully computed a second-order *ShockWave* is initialized and inserted in the

update directions from (x,y) / previous ray directions	update direction 1	updated direction 2
direction =0	(x+1,y)	
0 < direction < 45	(x+1,y)	(x,y+1)
direction = 45	(x+1,y+1)	
45 < direction < 90	(x+1,y+1)	(x,y+1)
direction = 90	(x,y+1)	
90 < direction < 135	(x,y+1)	(x-1,y+1)
direction = 135	(x-1,y+1)	
135 < direction < 180	(x-1,y)	(x-1,y+1)
direction = 180	(x-1,y)	
180 < direction < 225	(x-1,y)	(x-1,y-1)
direction = 225	(x-1,y-1)	
225 < direction < 270	(x-1,y-1)	(x,y-1)
direction = 270	(x,y-1)	
270 < direction < 315	(x,y-1)	(x+1,y-1)
direction = 315	(x+1,y-1)	
315 < direction < 360	(x+1,y-1)	(x+1,y)

Table 6.2: This table specifies update discrete directions. *i.e.*. the grid points which a *DiscreteWaveFront* at a grid point (x, y) must visit in the next iteration.

OrderedWaveFrontList module. Figure 6.1b illustrates the formation of a second-order shock in the initialization stage. Note that each second-order shock the launches two first-order *ShockWaves* in opposite directions.

4. Ordered Wavefront List: This module uses a sorting algorithm (heap) to order *Discrete WaveFronts* and *ShockWaves* based on increasing distance values.

5. Discrete Wave Propagation: The *Discrete WaveFronts* are propagated on the discrete grid by using discrete directions. In each step, a *Discrete WaveFront* at a *DiscreteGridPoint* visits (or updates) a number of neighboring grid points. An analytic distance value between each visited grid and the source *GeometricBoundaryModel* of the *WaveFront* is computed. Note that this computation is purely local since the propagated parameters of the source *GeometricBoundaryModels* are locally available. If the *DiscreteGridPoint* has not been visited or its current distance value is more than the newly computed value, the latter is stored there. In addition, the source of *WaveFront* and discrete directions, which specify the next *DiscreteGridPoints* that this *WaveFront* should visit, are updated at this *DiscreteGridPoint*. Similarly, if this *DiscreteGridPoint* is not already in the *OrderedWaveFrontList* the *WaveFront* and its distance value are inserted in the *QuenchedWaveFrontList* as a possible *Discrete WaveFront* source. Otherwise, the existing *WaveFront* and its distance value in this *QuenchedWaveFrontList* are modified with the new one. Moreover, the dis-

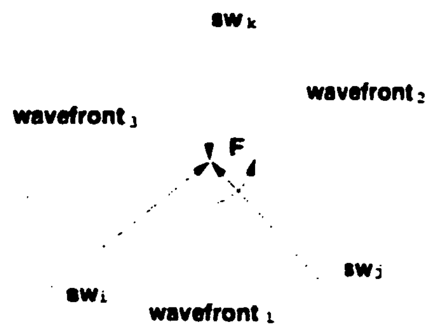


Figure 6.3: The shockwave sw_f at point F is a fourth order shock. The fourth-order shock is determined by all the neighbors of the incoming shockwaves, sw_i, sw_ij and sw_k.

crete directions associated with the current *WaveFront* are stored in this *DiscreteGridPoint* for further propagation. The initial discrete directions are summarized in Table 6.1. Once a *DiscreteGridPoint* is updated with a initial *DiscreteWaveFront* it is no longer initial and new directions are assigned to this *DiscreteGridPoint*: Table 6.2 specifies the updated directions from previous ones.

In analogy to *DiscreteWaveFront* initialization stage, each *DiscreteGridPoint* stores the *WaveFronts* that visit it. Specifically, when a *DiscreteWaveFront* visits a *DiscreteGridPoint*, its source information is stored in the *QuenchedWaveFrontList* of this grid. When a *DiscreteWaveFront* visits a grid point and there are other *WaveFronts* in the *OWFL* at that point a quench is signaled which launches the computation of a second-order shock.

6. Shockwave Initialization for Propagation: When a shockwave forms, but has not propagated, it is important to check whether it needs to be propagated. Since a fourth-order shock is characterized by one where all the branches have incoming velocities, its type can be determined by all the neighbors of the incoming *ShockWaves*, Figure 6.3. If the shockwave is a fourth-order shock, this point is marked as a fourth-order shock and is removed from the *OrderedWaveFrontList*. If the shockwave is not a fourth-order shock, its shock path is computed in this initial stage. Three main steps are involved in the determination of the shock path of a *ShockWave*:

(1) *The analytic computation of the shock path:* The shock path is determined by analytic formulas for the bisector of the *GeometricBoundaryModels* of the *ShockWave*. It should be observed that a *ShockWave* propagates only on a portion of this bisector curve. The valid bisector segment can be determined by finding the beginning and end

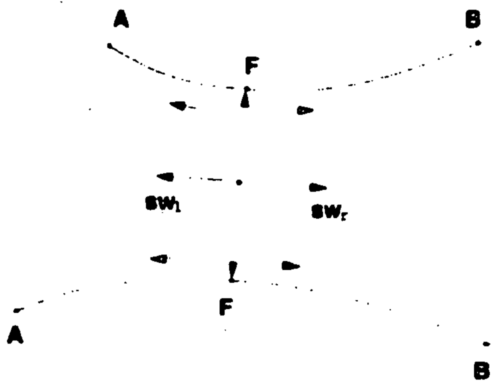


Figure 6.4: A second-second shock formed at P initiates two shockwaves, sw_l and sw_r, each moving in opposite directions. The neighboring shockwaves of each shockwave are determined from *GeometricBoundaryModels*. For example, one of its two neighboring shockwaves of the shockwave sw_r is determined by detecting the first shockwave on a propagation which takes place on the upper geometric boundary model, specifically from the foot point F to the boundary end point B.

points of the shock path. The beginning point of a shock path corresponds to a point where a *ShockWave* forms. The end point of a *ShockWave* is determined from the intersection of the shock path with its neighboring (left and right) *ShockWaves*. The end point is constantly updated as more information becomes available. For example, initially a wavefront typically has (but can have) no neighbors so that the end point is at infinity. However, as second-order shocks form, the analytic intersection leads to predicted junctions, thus updating end points. Again, these points may never reached.

(2) *Determining the neighboring ShockWaves of a ShockWave:* These are determined based on the formation type of a *ShockWave*. Specifically, (*i*) for a *ShockWave* formed from the center of a curvature maximum of a *GeometricBoundaryModel*, the neighboring *ShockWaves* are easily determined from the contact *ShockWaves* originated from the end points of this *GeometricBoundaryModel*; (*ii*) for a *ShockWave* formed from a second-order shock, the neighboring *ShockWaves* are determined from the *GeometricBoundaryModels* that formed the second-order shock as follows. Recall that when a second-order shock forms the *WaveFront* splits into two pieces, thus initiating two *ShockWaves* moving in opposite directions. A neighboring *ShockWave* is computed by moving on a *GeometricBoundaryModel* from the foot point of the second-order shock to the end point of the *GeometricBoundaryModel* in the direction of the *ShockWave* and by detecting the first *ShockWave* (or its current child *ShockWave*) as a

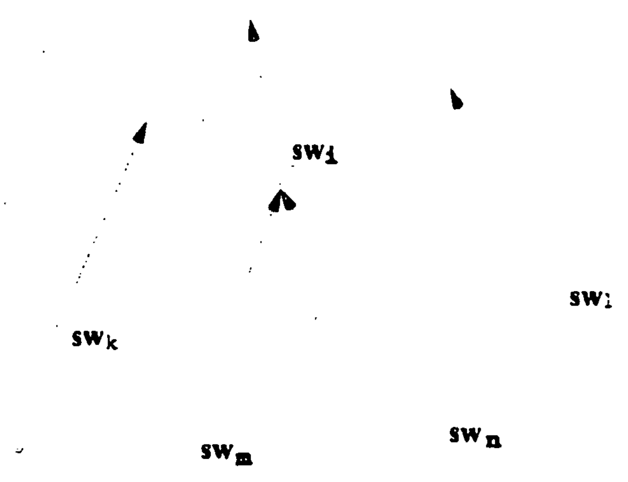


Figure 6.5: A *ShockWave*, sw_i is formed from the intersection of *ShockWaves*, sw_m and sw_n. Observe that the neighboring *ShockWaves* of sw_i are the left shockwave of sw_m, $lsw(sw_m)$ and the right shockwave of sw_n, $rsw(sw_n)$.

neighboring shockwave, Figure 6.4; (*iii*) for a *ShockWave* formed from a junction, the neighboring shockwaves are inherited from the parent *ShockWaves*. Specifically, suppose that a new *ShockWave*, sw_i is formed from the intersection of *ShockWaves*, sw_m and sw_n, Figure 6.5. Let the notation the $rsw(sw_m)$ indicate the right *ShockWave* of sw_m and, similarly let $lsw(sw_n)$ indicate the left *ShockWave* of sw_n. Then, we have

$$\begin{cases} rsw(sw_m) = sw_n; \quad lsw(sw_m) = sw_k \\ lsw(sw_n) = sw_m; \quad rsw(sw_n) = sw_l. \end{cases} \tag{6.1}$$

Then, the neighboring *ShockWaves* of sw_i will be

$$\begin{cases} lsw(sw_i) = sw_k \\ rsw(sw_i) = sw_l. \end{cases} \tag{6.2}$$

In addition, the neighboring shockwaves of $lsw(sw_m)$ and $rsw(sw_n)$ are modified accordingly, *i.e.*,

$$\begin{cases} rsw(sw_k) = sw_i \\ lsw(sw_l) = sw_i. \end{cases} \tag{6.3}$$

(3) *Determining the end point of a ShockWave:* This is determined from the intersection of the shock path with each of the neighboring shock paths. Suppose that a *ShockWave*, sw_i has the neighboring *ShockWaves* sw_l and sw_r. Let the notation I_{il} denotes the intersection point of *ShockWaves* sw_i and sw_l, Figure 6.6a. Similarly define I_{ir}, I_{lk},

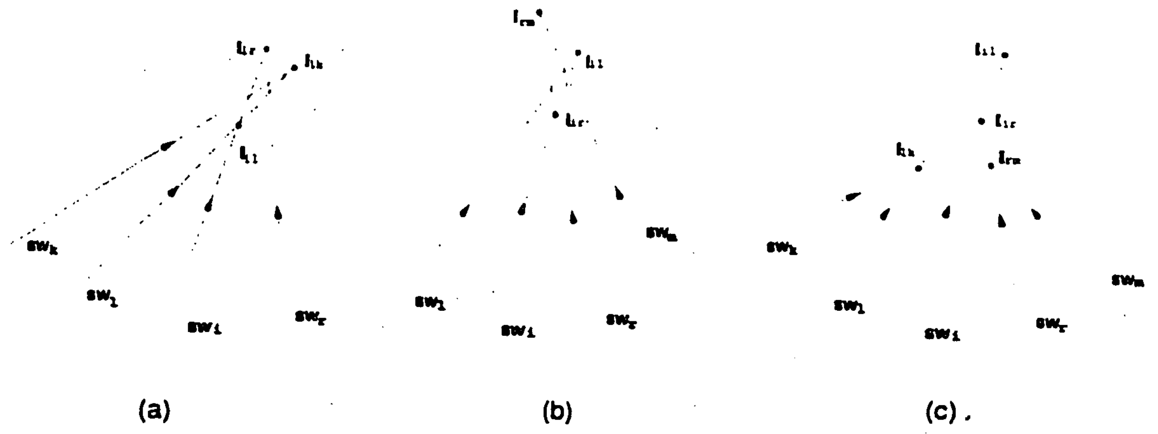


Figure 6.6: This figure illustrates the determination of an end point of a shockwave, sw_i whose neighboring shockwaves are sw_l and sw_r. (a) end point from the intersection of sw_i and sw_l, (b) end point from the intersection of sw_i and sw_r, (c) no valid end point is detected at this moment.

I_{rm}, *etc.* Let a point, E_i denotes the correct end point of the *ShockWave* sw_i. This point forms from its intersection with one of the neighboring shockwaves as follows. The determination of the end point of the shockwave, sw_i is based on the intersection points I_{il} and I_{ir}, and the distance values, $dist(I_{il})$, $dist(I_{ir})$, $dist(I_{lk})$, and $dist(I_{rm})$. Specifically, the end point of the *ShockWave*, sw_i will be one of the three possible cases illustrated in Figure 6.6:

(1) The shockwave sw_i may first intersect with the left shockwave, sw_l, *i.e.*, $dist(I_{il}) < dist(I_{ir})$ and $dist(I_{il}) < dist(I_{lk})$ (Figure 6.6a), implying that

$$\begin{cases} E_i \leftarrow I_{il} \text{ and } & dist(E_i) \leftarrow dist(I_{il}) \\ E_l \leftarrow I_{il} \text{ and } & dist(E_l) \leftarrow dist(I_{il}) \end{cases} \tag{6.4}$$

(2) The shockwave sw_i may second intersect with the right shockwave, sw_r, *i.e.*, if $dist(I_{ir}) < dist(I_{il})$ and $dist(I_{ir}) < dist(I_{rm})$ (Figure 6.6b)

$$\begin{cases} E_i \leftarrow I_{ir} \text{ and } & dist(E_i) \leftarrow dist(I_{ir}) \\ E_r \leftarrow I_{ir} \text{ and } & dist(E_r) \leftarrow dist(I_{ir}) \end{cases} \tag{6.5}$$

(3) The shockwave sw_i may not intersect with any of the neighboring *ShockWaves*, sw_l, and sw_r, *i.e.*, if $dist(I_{il}) > dist(I_{lk})$ and $dist(I_{ir}) > dist(I_{rm})$ (Figure 6.6c)

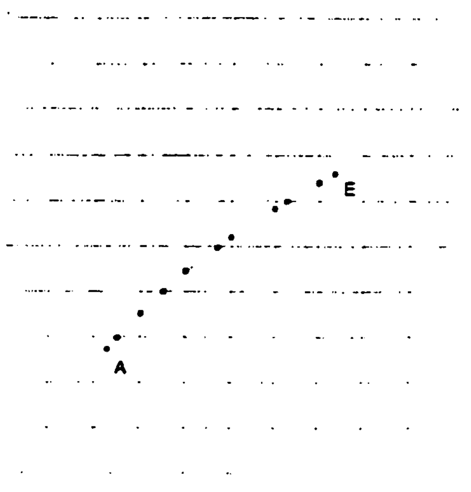


Figure 6.7: This figure illustrates the propagation of a *ShockWave* from point A to point E. The propagation takes place on the intersections of grid with its shock path. grid crossings.

$$\begin{cases} \text{no end point is valid} \\ E_i \leftarrow \mathit{null} \qquad \text{and} \qquad \mathit{dist}(E_i) \leftarrow \text{large number} \end{cases} \tag{6.6}$$

7. **Shockwave Propagation:** A *ShockWave* propagates on the intersections of grid lines (horizontal and vertical) with its shock path until it reaches to its end point or leaves the image domain. Figure 6.7 illustrates the shock path of a *ShockWave* which is formed at a point A and which terminates at point E. In addition. a *ShockWave* typically intersects with horizontal and vertical grid-lines. At these grid-line samples we store its *GeometricBoundaryModels* to the *QuenchedWaveFrontList* of its immediate left and right (up and down) grid points, Figure 6.8. When a *WaveFront* visits a *DiscreteGridPoint* and there are other *WaveFront* in the *QuenchedWaveFrontList* at that point a quench is signaled which launches the computation of a second-order shock. When a *ShockWave* reaches its end point, it is terminated and is removed from the *OrderedWaveFrontList*. From this end point, if a new *ShockWave* form, it is inserted to the *OrderedWaveFrontList*. However, this new *ShockWave* from this end point will not be initialized if the end point itself is not valid. Specifically, suppose that a *ShockWave*, sw_i at a point A moves to its end point B which is the intersection of sw_i and sw_r, Figure 6.9. A new *ShockWave* forms from this end point B, which will be initialized when it is chosen for propagation. However, a *ShockWave* sw_n might form from the intersection of sw_l and its neighbor sw_k at a place where its distance is greater than the distance of *ShockWave* sw_i at the point A and less than the distance at point B. In addition, this *ShockWave* sw_n might intersect with the *ShockWave* sw_i at

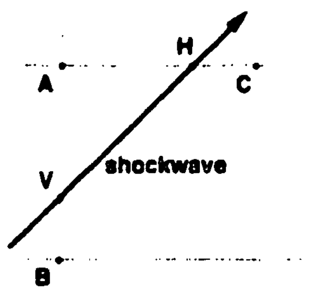


Figure 6.8: A *ShockWave* propagates on the intersection of the grid with its shock path. When the *ShockWave* intersects with a vertical grid (V) it stores the *WaveFront* information to the *QuenchedWaveFrontList* of its immediate up (A) and down (B) *DiscreteGridPoints*. Similarly, a *ShockWave* on a horizontal grid (H) stores the *WaveFront* information to the *QuenchedWaveFrontList* of its immediate left (A) and right (C) *DiscreteGridPoints*.

a point between A and B, thus making point B invalid end point, thus *ShockWave* should not initialized from this point.

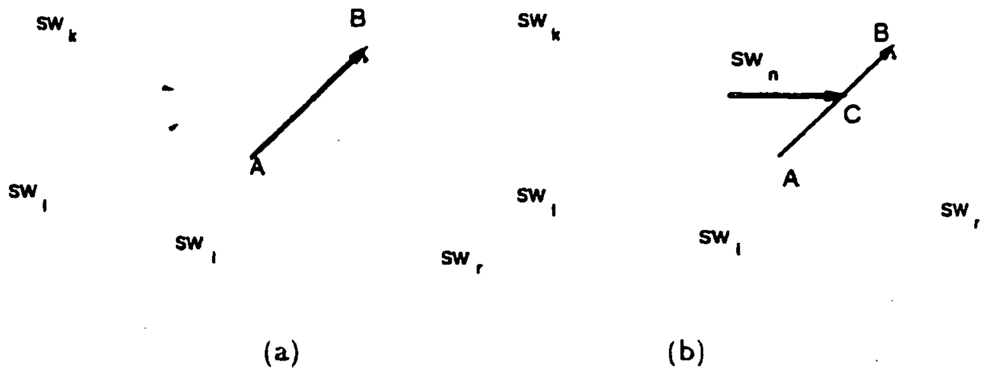


Figure 6.9: (a) A *ShockWave* sw_i propagates from point A to its end point B which is the intersection of its shock path with the shock path of *ShockWave* sw_r. (b) The new *ShockWave*. sw_n formed from sw_l and sw_k intersect with *ShockWave* sw_i at the point C. Observe that the junction point B is no longer valid and the *ShockWave* initialized from this point has to be deleted.

6.4 Computational Complexity of The Wave Propagation Algorithm

The computational complexity of our algorithm depends on the number of boundary elements N, and the number of discrete grid points in the image domain MxM. We study its complexity by dividing the algorithm into a number of independent modules and considering each separately, Figure 6.2: (I) discrete wavefront initialization; (II) formation of contact and A_3 initial shockwaves; (III) second-order shock computation; (IV) discrete wavefront propagation; (V) initialization of shockwaves; and (VI) shockwave propagation [1].

(I) Discrete wavefront initialization: In this module, the discrete grid points that enclosing the geometric boundary models are marked as the initial wavefront. The computational complexity of this module depends on the number of geometric boundary model, N and the image size M. Observe that the length of a boundary model cannot be greater than the diagonal of the image domain, *i.e.*, $M\sqrt{2}$ pixels. Thus, each boundary model in a pixel initiates discrete waves at each corner of the pixel where it lies, thus requiring at most $4M\sqrt{2}$ *initial update*. An initial update of the wavefront requires several steps: (i) the distance between each point and each of the N geometric boundary models must be computed. The worst complexity of this step is therefore $O(N.M)$; and (ii) this computed distance must be compared with previously stored distance values. If the computed distance value is not smaller than previous values, no additional computations are required. However, if the computed distance value is smaller than previous values then (the worst complexity of this step is $O(N.M)$): (iii) discrete grid directions must be computed for further propagation. This implies the the vector between the updated point and its foot point on the geometric boundary model be computed. The worst case complexity of this step is $O(N.M)$. In addition, (iv) this distance value must be inserted in the ordered wavefront list, requiring $O(N.Mlog(N.M))$ comparisons [2]. Finally, (v) the wavefront at this grid point must be saved. The worst case complexity of this step is $O(N.M)$. Thus, the worst case complexity of this module is $O(N.Mlog(N.M))$.

(II) Formation of Contact and A_3 Shockwaves: The worst case complexity of this module is $O(N)$ since each boundary model can initiate at most $4N$ contact shockwaves and a constant multiple of N A_3 shockwaves.

(III) Second-Order Shock Computation: A second-order shock computation is only required when two boundary models have visited the same grid crossing during the wave propagation stage [3]. The worst case computational complexity corresponds to when all the

[1]It should be observed that the computationally complexity of ordering discrete wavefronts and shockwaves are included in "discrete wavefront propagation" and "shockwave propagation".

[2]The complexity of sorting a list of N element is $O(Nlog(N))$.

[3]For the space f geometric models up to circular arcs this requires, in the worst case, solving the roots of a quadratic equation.

boundary elements are in or will eventually visit the same grid point, implying a roughly circular arrangements of source models. In this case, ($\frac{N}{2}$) computations are required leading to the worst case complexity of $O(N^2)$. Note, however, that when edges are uniformly distributed, one per pixel at most, then the computation is optimal and of order $O(N)$.

(IV) Discrete Wavefront Propagation: In the worst case, each grid point is updated by every boundary model in the image domain, resulting on $O(N.M^2)$ updates. Generically, however, each pixel is updated by three models resulting in $O(M^2)$ updates. An update requires that : (i) the distance between a point and a geometric boundary model be computed. The worst case complexity of this step is $O(N.M^2)$. (ii) the computed distance be compared to previous distance values. The worst case complexity of this step is $O(N.M^2)$. If the computed distance value is smaller than previous values: (iii) the new distance value should be inserted in the ordered wavefront list, which requires $O(N.Mlog(N.M))$ comparisons. Finally, (iv) the wavefront at this grid point should be saved, which requires at most $O(N.M^2)$ computations. If the computed distance value is larger than previous value, no additional computations are required. The worst case is possible when all boundary elements are located in a single pixel. Thus, the worst case complexity of this module is $O(N.M^2)$, therefore with typical complexity of $O(M^2)$.

(V) Initialization of Shockwaves: Held [81] proved that there are a maximum of $(6N-6)$ bisectors for N boundary segments. A shock path computation requires (i) analytically computing bisectors from two boundary models of the shockwave, *i.e.*, only the coefficients of a bisector curve are computed from a given formula; (ii) determining neighboring shockwaves, and (iii) computing the intersection of each bisector with the bisectors of the left and right shockwaves to find junction points. Thus, this module requires at most $O(N)$ *shock path computations.*

(VI) Shockwave Propagation: A shockwave propagates on the intersection of a grid with its bisector curve. The computation of a grid crossing, namely, the intersection of a grid line and the shock path, requires finding the roots of a polynomial, *e.g.*, quadratic polynomial if the boundary models are restricted to lines and circular arcs. In the worst case, the length of a shock path is of order $O(M)$. Since there are $O(N)$ shocks, the worst case complexity of retrieving all grid crossings shocks is $O(N.M)$.

Overall Complexity: Using the above computations, the overall worst case complexity of the algorithm is $O(NM^2)+O(N^2)+O(N.Mlog(N.M))$. This is achieved when conditions

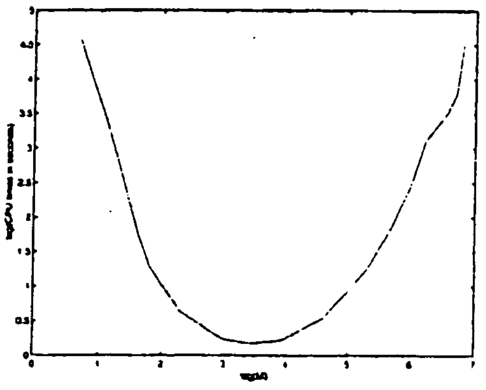


Figure 6.10: This figure depicts the graph of the CPU times (s) spent by the wave propagation algorithm in constructing the symmetry representation of the edge map (330 edge elements) for varying image sizes.

for worst case complexity for each module are met, which is highly unlikely. The average case complexity is far below this level. Since our results are exact regardless of the underlying grid size M^2, we can optimize the algorithm's expected performance for a given N. Observe that the algorithm behaves differently as its two extremes, namely, when $M \rightarrow \infty$ and when $M \rightarrow 1$, as N is held to constant. First, the condition $M \rightarrow \infty$ implies increasing the underlying grid resolution, which also increases computational time to infinity dominated by $O(M^2)$; this must be clearly be avoided! Second, as $M \rightarrow 1$ the wave propagation aspect is reduced, but at the expense of increasing bisector computations, dominated by $O(N^2)$ complexity. In the limit, the algorithm for obtaining skeletons becomes independent of an underlying grid! Thus, the optimal M lies somewhere between these extremes, as Figure 6.10 illustrates. We will return this in Section 7.2.3 as in the context of comparison to Voronoi Diagrams..

6.5 Results and Discussion

Figure 6.11 and 6.12 illustrate the formation and propagation of shockwaves. Note that since *all* shockwaves are computed exactly, pruning is required. This is accomplished by pruning a branch (not a single shock), as a transformation of the symmetry map [216]. Figure 6.13 and 6.15 illustrate examples of closed shapes arising in various applications. Figure 6.17 and 6.18 are the symmetry map of the edge map corresponding to a collection of curve segments. Figure 6.19 illustrates the symmetry map of a synthetic edge map containing junctions. The symmetry map of an edge map of a real image is shown in Figure 6.20. The main point is that the symmetry map of an edge map, when exact and robust

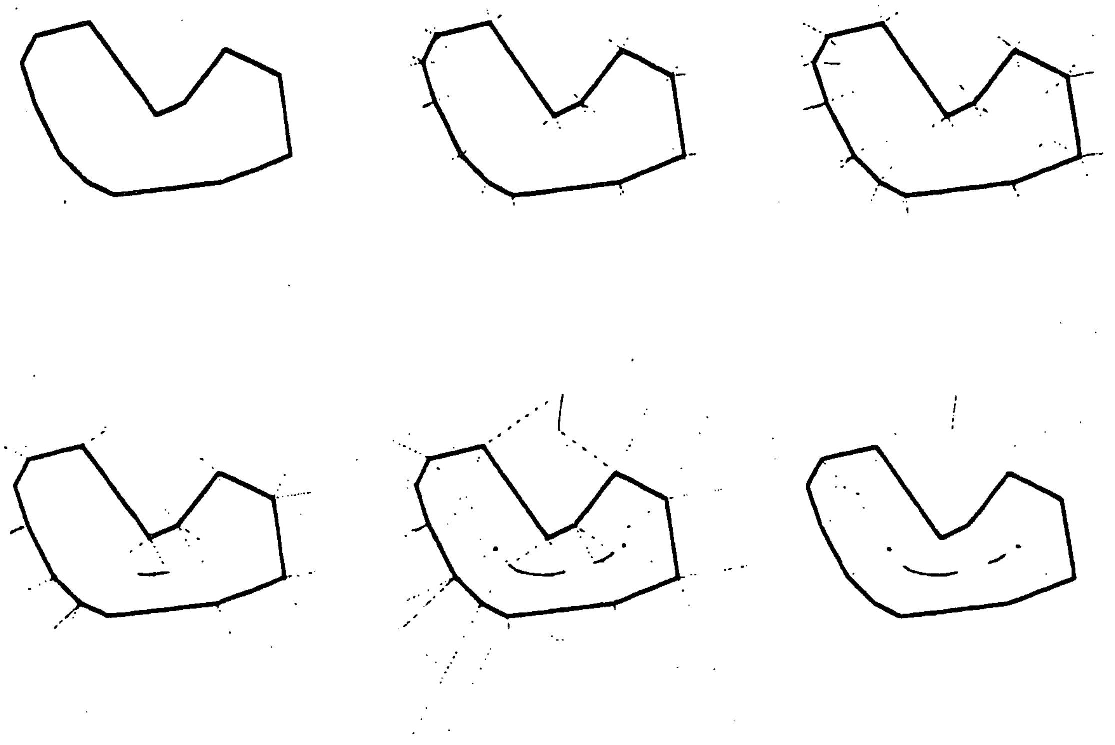


Figure 6.11: This figure illustrates the formation and propagation of shockwaves on discrete domain from a set of unorganized edge segments.

computations are available, can be used as an appropriate language to express grouping operations by defining transformations such as smoothing, gap completion, partitioning, removal of spurious elements, *etc.* towards forming object hypotheses [216].

Our contributions are (*i*) exact construction of Voronoi diagrams of the objects represented by N curve segments; (*ii*) our algorithm has low order complexity due to the wavefront propagation; (*iii*) The algorithm is based on local operations, thus making it extremely important in symmetry transforms represents grouping operations [216, 75]; (*iv*) no post-processing is necessary for tree-like graph representation of shapes; (*v*) applicable to curve segments with possible intersections with other curve segments, and contour with junctions; (*vi*) extensible to 3D; (*vii*) exact construction of wavefronts moving with constant speeds, *e.g.*, exact distance transforms.

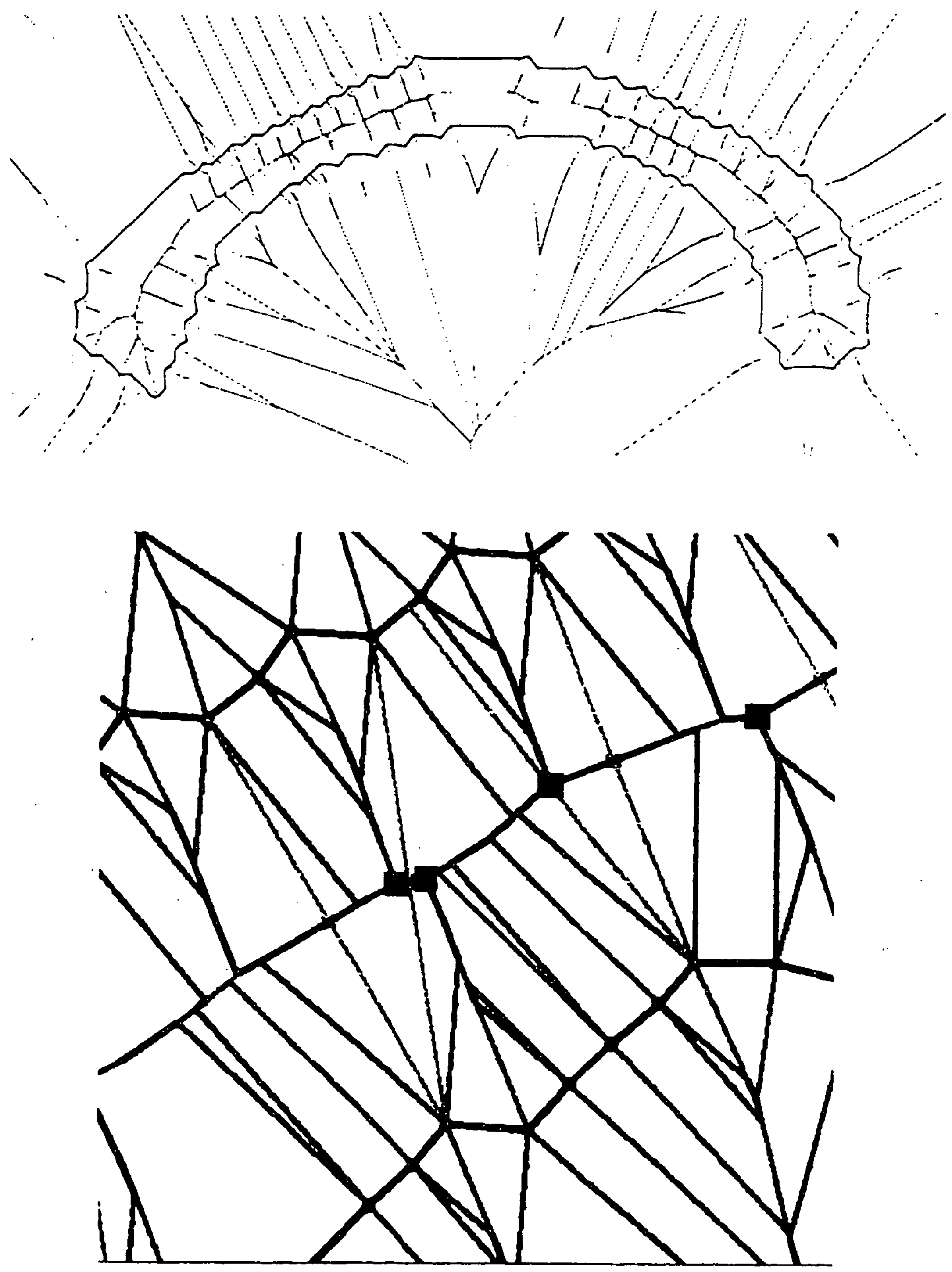


Figure 6.12: A discrete shape and its detected shock waves. Green, blue and red identify regular, semi-degenerate, and degenerate shockwaves, respectively. Dashed lines/grey are the end point rays. (Bottom:) A zoomed portion to illustrate the shockwaves on the simulation grid (yellow). Direction of propagation is not shown for simplicity. Filled squares denote fourth-order shocks. Light blue lines illustrate the formation of second order shocks. Observe the formation of several shocks in a single pixel.

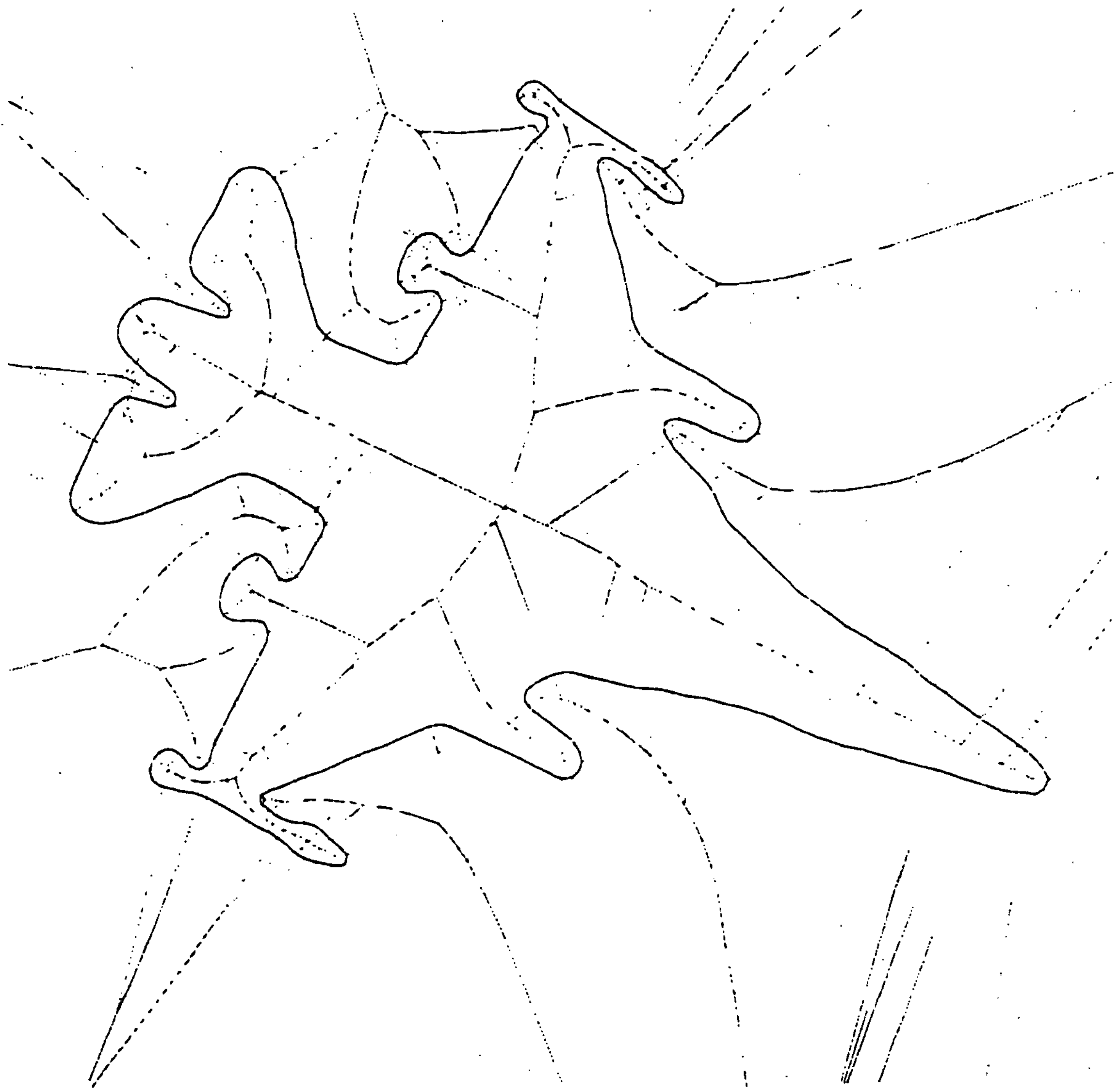


Figure 6.13: This figure illustrates the computation of shocks from the boundary represented as a set of unorganized edge segments (which appear attached at low resolution) segments. Note that no edge ordering was necessary.

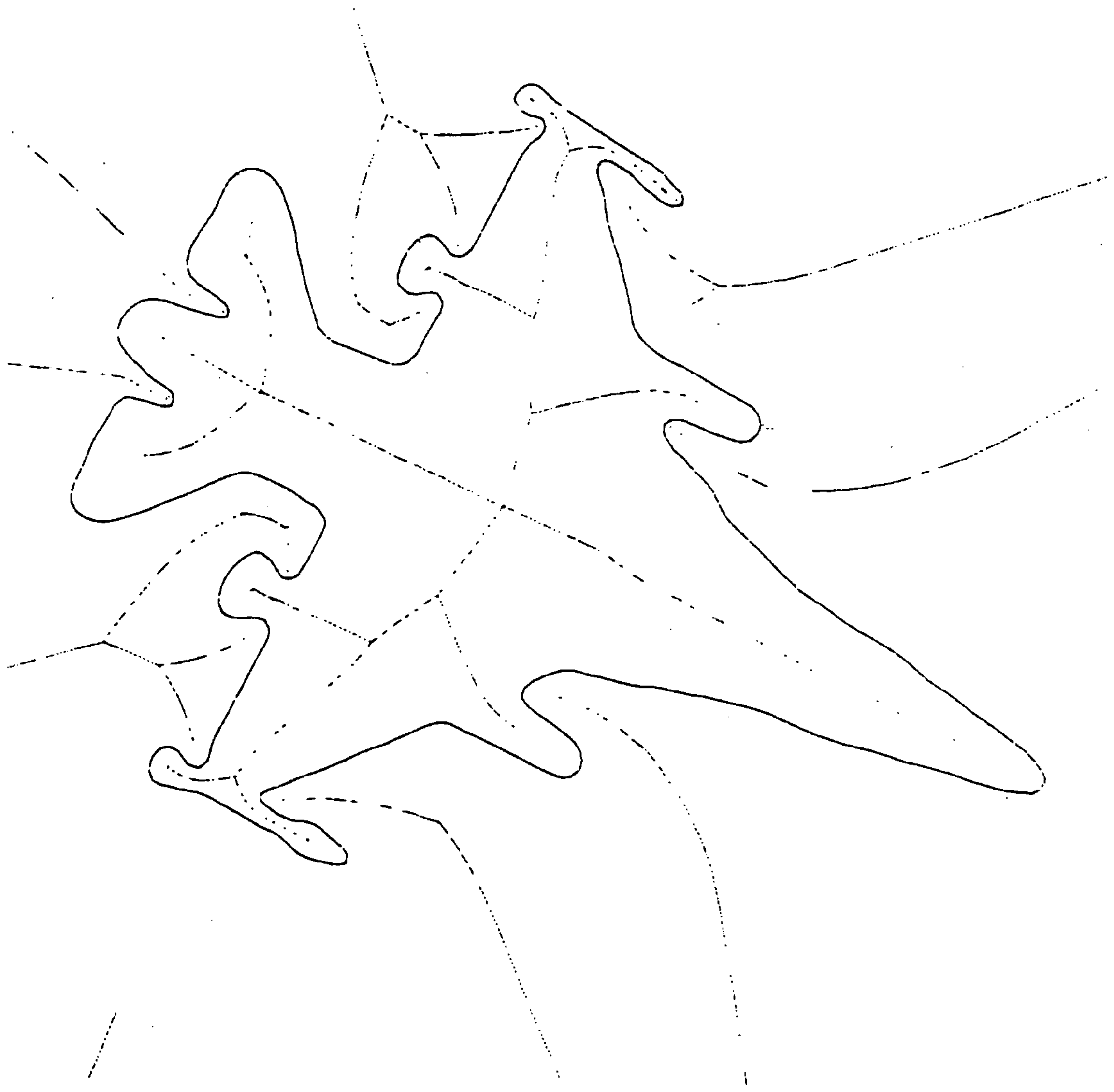


Figure 6.14: Pruned shocks of the previous example in Figure 6.13 to remove discretization effects.

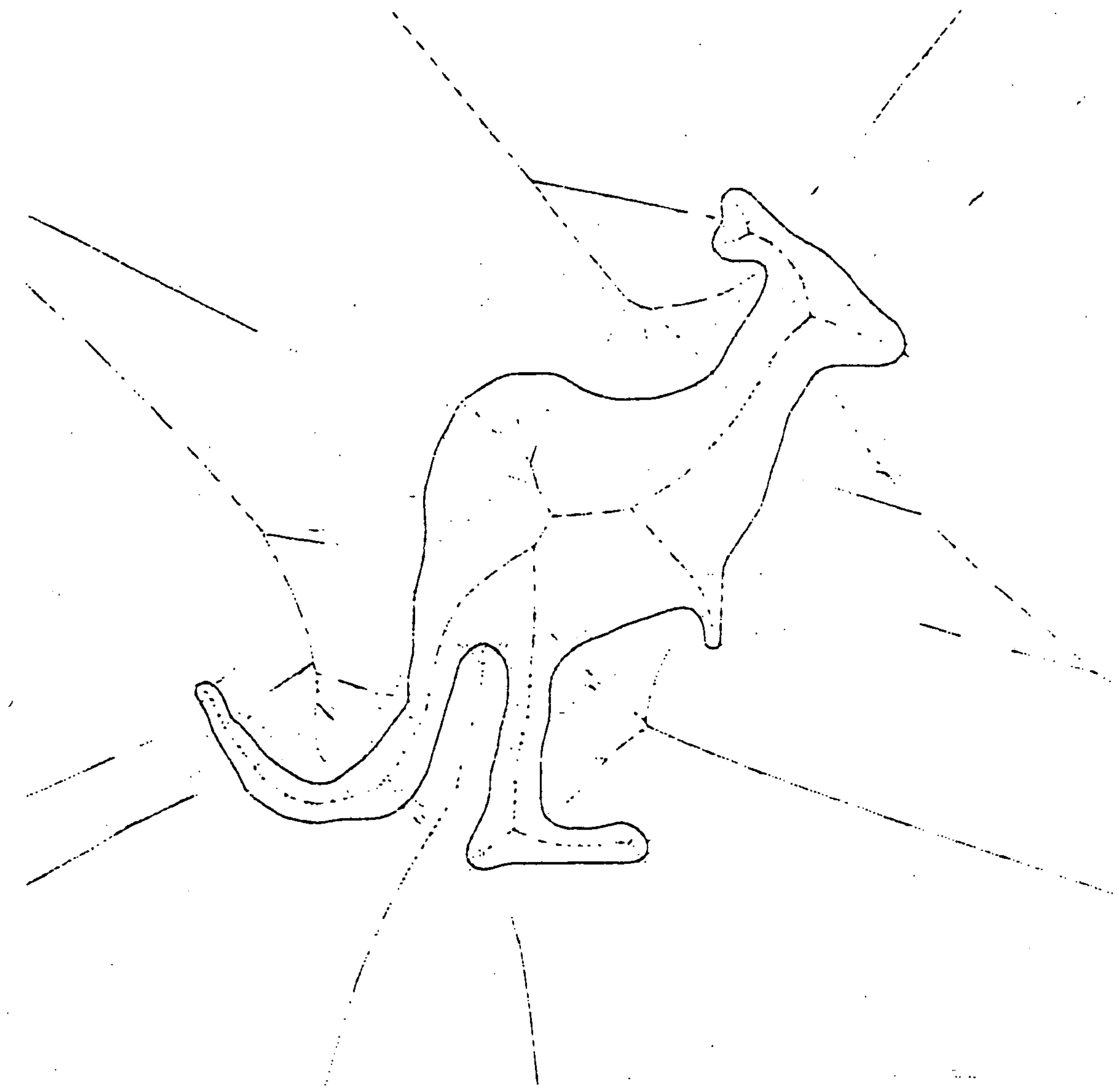


Figure 6.15: This figure illustrates the computation of shocks from the boundary represented as a set of unorganized edge segments (which appear attached at low resolution) segments before Note that no edge ordering was necessary.

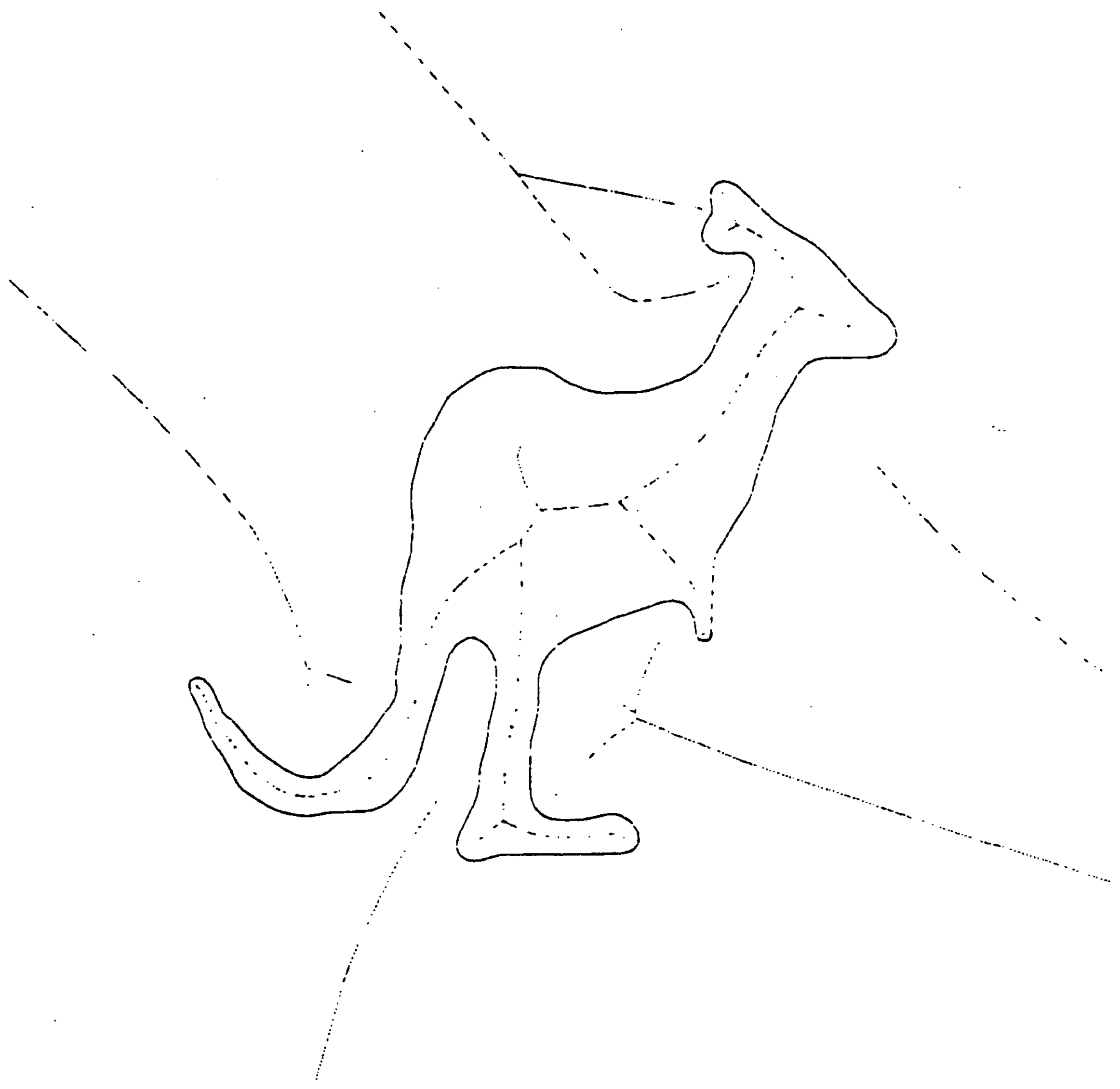
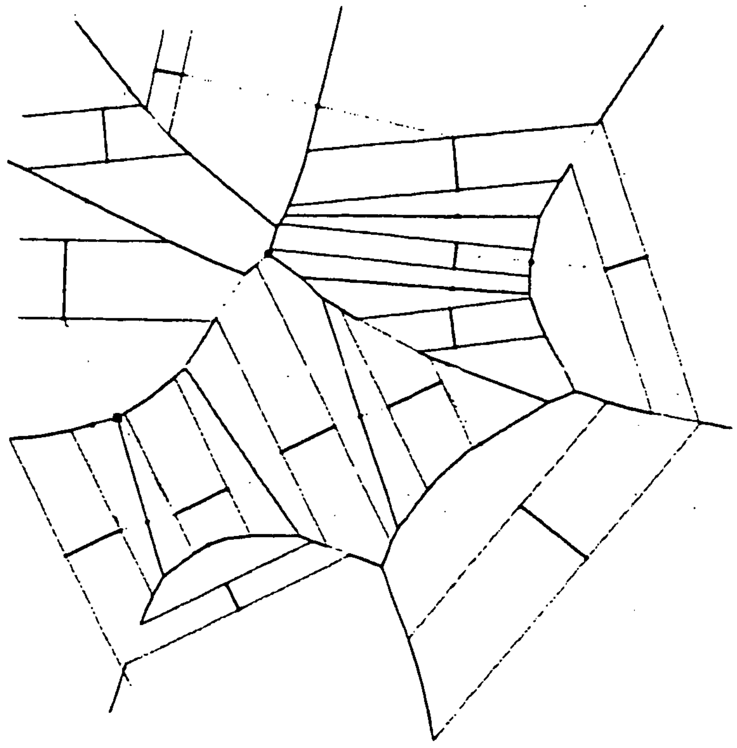


Figure 6.16: Pruned shocks of the previous example in Figure 6.15 to remove discretization effects.



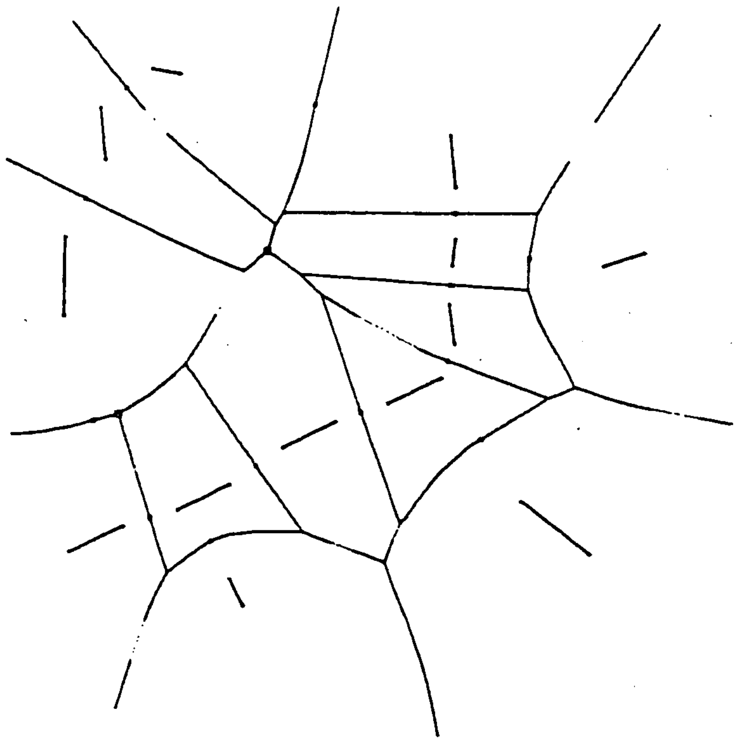


Figure 6.17: This figur shows shocks obtained from a set of curve segments before and after pruning. Note that *red* segments are degenerate segments which suggest gr uping of their edge el ments.

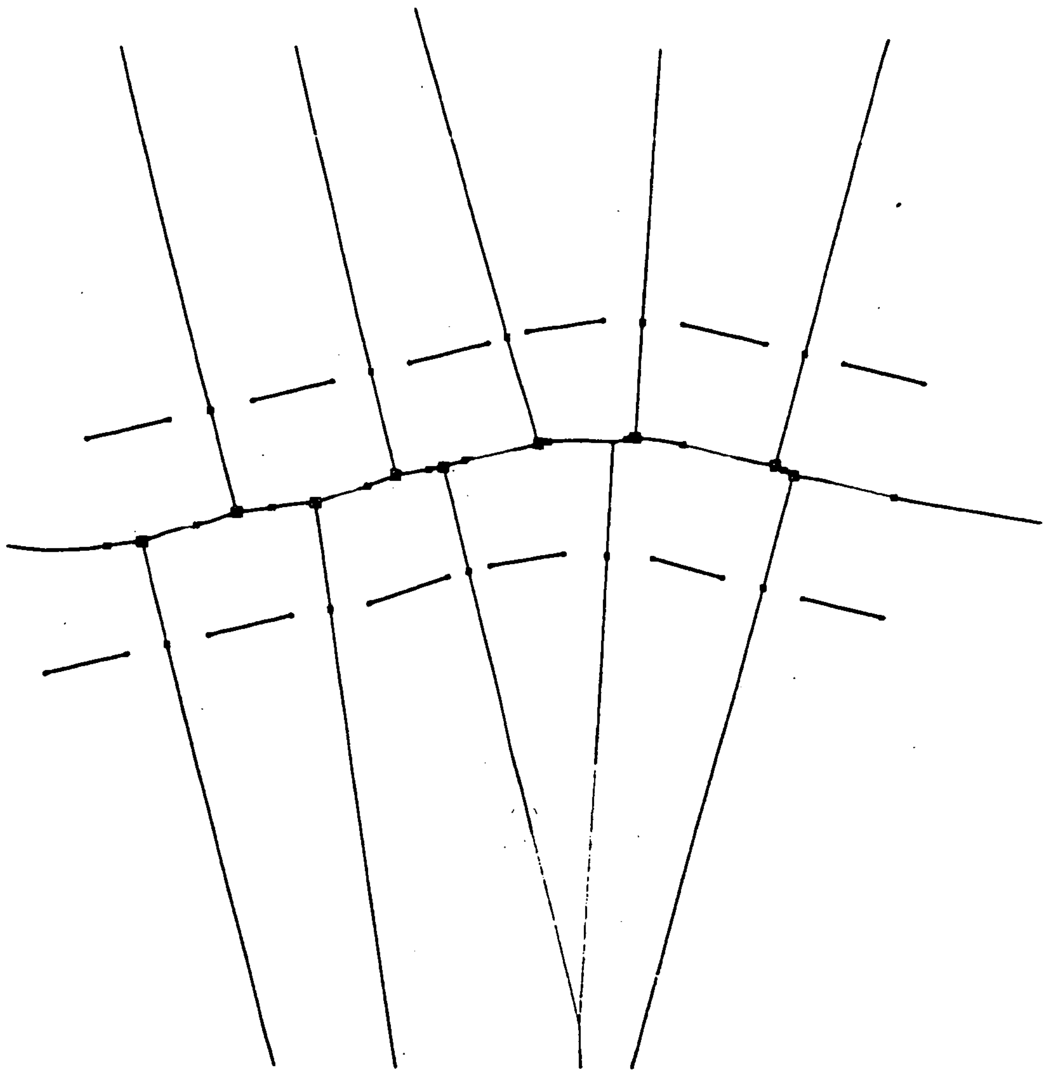


Figure 6.18: This figure illustrates the computation of a symmetry map for an edge map. Observe that the red branches (degenerate) indicate grouping of edge segments which can be implemented by locally modifying the symmetry map.

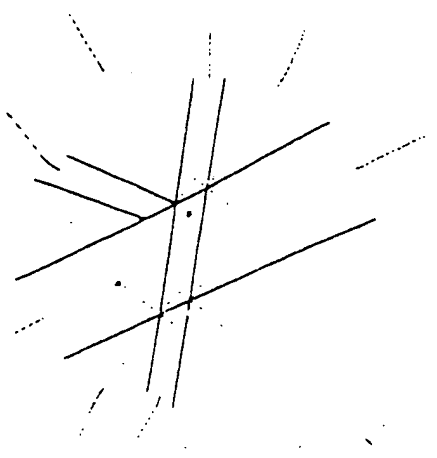


Figure 6.19: The symmetry map of a synthetic edge map containing several junctions.

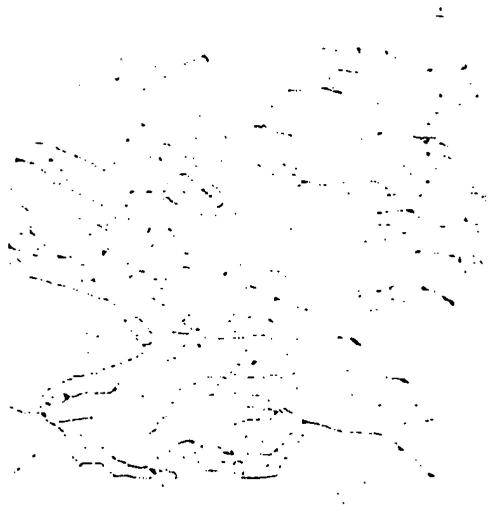


Figure 6.20: (a) Original gray scale image and (b) The symmetry map of the edge map obtained by applying Canny's edge detection algorithm [32] t the image in (b).

Chapter 7

Wave Propagation, Curve Evolution and Voronoi Diagrams

In this chapter, we review two previous symmetry detection approaches, one based on curve evolution and the other based on computing the Voronoi Diagram. Note that since our approach is based on a combination of analytic curve evolution and bisector computation, it is important to relate our approach to each.

7.1 Curve Evolution

In this method, object boundaries are deformed by partial differential equations, and singularities, or shocks, formed during the propagation are computed by detecting the discontinuities on the deforming curve [176, 112, 196, 209]. Kimia *et al.* [106, 112] described shape as the collection of four types of shocks (singularities) which are formed in the course of deformations of shape in the reaction-diffusion space, with reaction representing constant flow and diffusion representing curvature flow. These deformations are implemented via the *curve evolution* paradigm by embedding the curves as the zero level set of a surface evolving by a corresponding reaction PDE. Siddiqi and Kimia [196] used this *curve evolution* framework to detect the shocks of evolving curve with subpixel accuracy. In this method, first-order shocks (discontinuities along the evolving curve) are detected by modeling the zero level set of the evolving surface by piecewise circular arc segments, recovered using a nonlinear geometric interpolation method, GENO [198] giving subpixel accuracy. These shocks are then grouped together to form shock branches. Note that GENO models have to be recovered at every step of the deformation process, thus representing additional computational complexity. The second-order (topological changes on the surface) and fourth-order shocks are detected from the deforming surface by determining the critical points at each stage of evolution using ENO methods, resulting in robust and accurate (subpixel) detection of shocks. The third-order shocks are detected when two evolving contours (represented by piecewise circular arcs) collide with each others at the same time for all the points of the contours. The results are robust and accurate across a range of shapes.

However, this approach has the following shortcomings: (i) it cannot be applied to open contours, T, Y, and X junctions, *etc.*, as frequently present in edge maps. This is mainly because the representation is based on an embedding surface, which requires a closed curve, which in turn assumes the availability of a *segmented shape* from gray level images: (ii) it is computationally inefficient due to the additional embedding dimension for implementing curve evolution and the extraction of an explicit model of the curve from the implicit one at each stage: (iii) grouping of shock points is necessary since exact curve evolution is not possible via this method and noise in the discrete domain gives rise to numerous isolated singularities.

In a related curve evolution approach, Tari and Shah [210] define symmetries as the curvature maxima of level sets which have been constructed consistently with an edge strength functional. However, in this approach, shock branches are disconnected or sometimes cross each other. In addition, in both Siddiqi and Kimia's, and Tari *et al.*'s work the notion of scale and detection are inter-mixed in the process, because some diffusion (smoothing) is needed to obtain stable results.

The main advantage of our wave propagation approach over curve evolution can be summarized as: (i) shocks of open contours with possible junctions can be detected with application to edge maps of images; (ii) no additional computational cost due to an embedding surface; (iii) no subsequent grouping operation is required since the shocks are detected exactly and grouped in the process; (iv) no post-processing is necessary to generate graph representation of shapes. (v) notions of symmetry detection and scale space representation are separated.

7.2 Computational Geometry

In computational geometry, it has been shown that the Voronoi Digram of a shape, is closely related to its symmetry set [161, 16, 147]. In this section, we briefly review algorithms to generate Voronoi Diagrams and compare them to wave propagation.

7.2.1 Voronoi Diagrams of Discrete Objects

The Voronoi Diagram (VD) extracts the symmetry set of a set of points. The VD divides the plane into convex polygons based on the nearest-neighbor rule which states that each point is associated with the polygon closest to it. Numerous techniques to generate the Voronoi Diagram of a discrete set of points have been proposed. We can roughly divide them into three main categories: (*i*) *incremental algorithms* start with a single point and proceed by incrementally inserting a new point into the diagram and modifying the diagram [150, 207]; (*ii*) *divide-and-conquer* techniques partition the point sets into two subsets of approximately equal size. The Voronoi Diagram of each subset is recursively computed and the resulting diagrams are then merged [161, 119, 148]; (*iii*) *sweepline-based* methods convert the static problem of computing a Voronoi diagram to the dynamic problem of maintaining the cross section of the diagram with a straight line [161, 16]. Specifically, a line is swept across the plane from bottom to top by constructing the Voronoi Diagram. The main advantage of VD of discrete objects over other symmetry detection algorithms is the efficiency of the VD algorithm and the accuracy of results (since exact VD of discrete points can be constructed rather easily.). The main issues in comparing VD algorithms are the worst case computational complexity and average performance in terms of CPU time.

A Comparison of wave propagation and VD: In this paper, we compare the performance of our algorithm with the VD algorithm of Ogniewicz [147] for objects whose boundaries are represented by a set of discrete points. This VD algorithm is based on a combination of incremental and divide-and-conquer techniques. The efficiency and availability of this approach have made it popular among computer vision researchers. Thus, the speed of our approach is compared with this approach, since this approach is optimized to run relatively fast.

Several points must be taken into account for this comparison. First, this VD algorithm takes as input a discrete set of *points* while our algorithm takes as input a set of curve segments and points, thus while it does not take advantage of the point nature of the source,

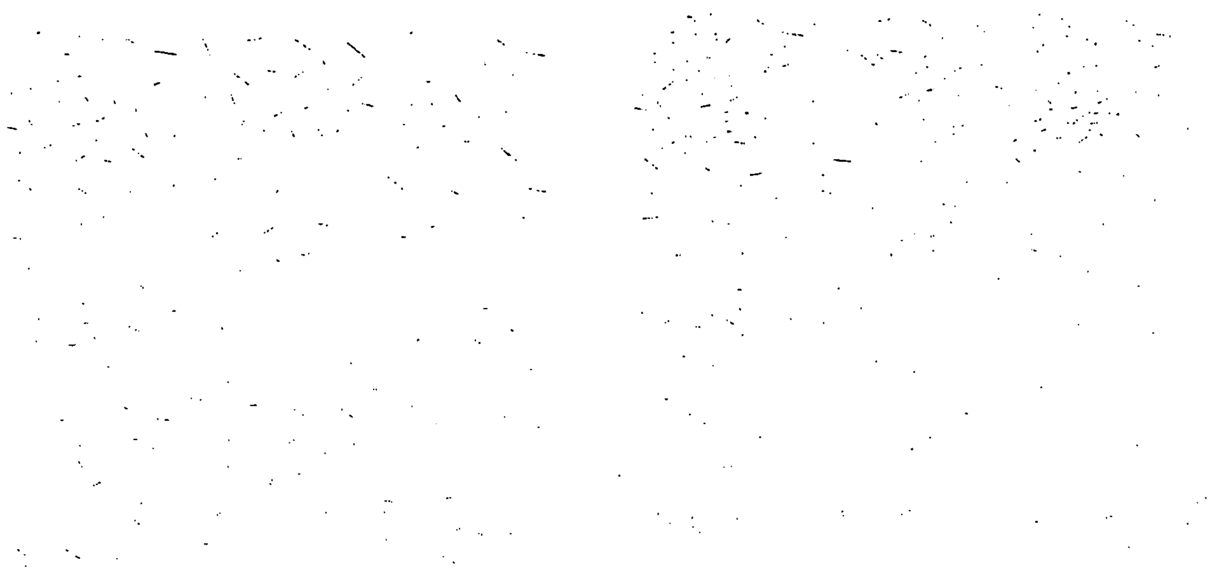


Figure 7.1: This figure illustrate an edge map with 330 elements and its symmetry representation. This example is used in the performance comparisons between our approach and Ogniewicz technique.

it can operate on a richer descriptional of the input. Second, our algorithm's complexity is dependent on the size of the grid. The optimal choice of grid size is in turn dependent on the number of edge elements. In contrast, the VD algorithm does not itself depend on the grid size, but the preprocessing step required to obtain a discrete set of points from an image (via tracing) does depend on the grid size, but only to a very small extent. There is, however, a minimum grid size required to capture small edge elements. Finally, we stress that our algorithm has not yet been optimized for efficiency. We now compare these approaches on two examples.

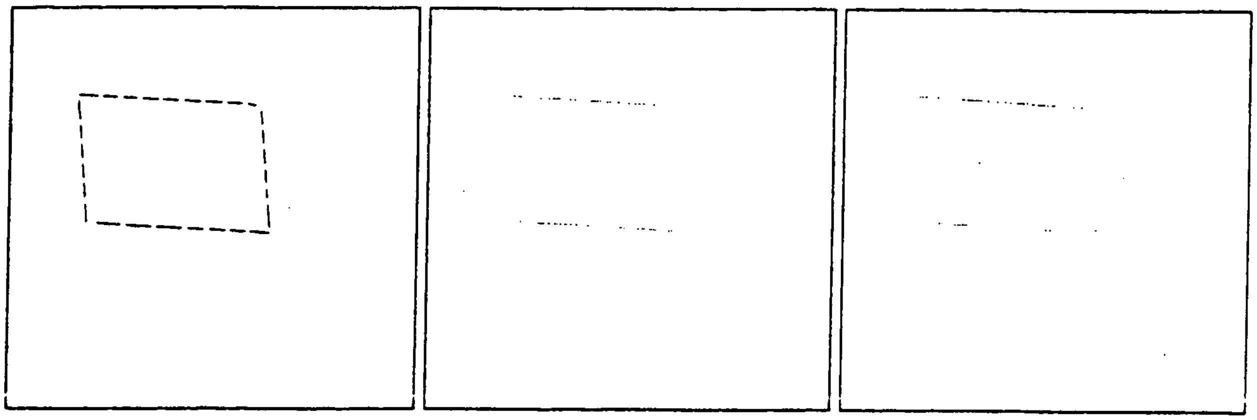
The first example contains 330 small edge elements, Figure 7.1. Table 7.1 shows the CPU times in seconds for these two approaches on a SUN SPARC 20 for various grid sizes, and graphically depicted in Figure 6.10 Observe that our algorithm exhibits two extremes. In one extreme, the speed of our algorithm increases drastically when the image size goes beyond 400x400. This is because when the image size M increases the discrete wavefront propagation aspect is intensified, such that the $O(M^2)$ minimum complexity becomes dominant in CPU time consumption. On the other extreme, when the image size M goes to one, *i.e.*, when a single large pixel embeds all the edge elements, a second-order shock computation is required from the each pairs of edge elements. Thus, N^2 second-order

image sizes / Approach	1x1	5x5	10x10	25x25	50x50	100x100	200x200	400x400	800x800
Wave Propagation	248.88	5.58	1.86	1.18	1.24	1.72	3.42	11.57	44.14
Ogniewicz's Approach (500x500)					0.85				

Table 7.1: This table shows the CPU times in second that are used by the wave propagation algorithm and the Ogniewicz's technique for detecting symmetry set of the edge map in Figure 7.1. The quality of results in the wave propagation method do not depend on the size of image domain. The comparable results are obtained by the Ogniewicz technique for a image size greater than 500x500 to generate sufficient accuracy.

shock computations dominate the time consumption of our algorithm. Thus, this extreme is prohibitive as well footnoteThis extreme is infact of the same order of complexity,and better than the method proposed in [166].. This is confirmed by the experimental plot, Figure 6.10, which shows that the optimal grid size is roughly 25x25. Observe that the minimum is fairly flat such that grids for 25x25 to 50x50 can be interchangeable used. In general, the optimal resolution should avoid repeative cases of multiple edge elements within the same pixel since the algorithms' accuracy is not dependent on the size of the grid. Thus, the optimal image size is dependent on the number and distribution of edge elements in the image domain. Specifically, assuming that the edge elements are uniformly distributed in the image domain, we can choose the number of grid points to be the minimum that typically produces one edge element per pixel, roughly $M = \sqrt{2N}$. As seen from Table 7.1, our wave propagation algorithm can perform as good as Ogniewicz's algorithm for the images containing uniformly distributed small edge elements.

In the second example, we consider an edge map, Figure 7.2a where clusters of edge elements can be naturally grouped into larger contour segments, Figure 7.2b as advocated by many approaches [188, 4, 156, 78, 82, 234, 224]. We will also present an approach based on symmetry transforms to group edge elements, Chapter 8. The main point is that groping reduces the number of edge elements, thus drastically improving efficiency. In our example, there are only four edge elements left after the grouping operations are performed. The speeds of our approach and Ogniewicz's approach are given in Table 7.2. As expected, our approach performs well since the number of edge elements are reduced



(a) (b) (c)

Figure 7.2: (a) an edge map containing small boundary segments (b) line segments obtained from a grouping process (c) their symmetry representation extracted by the wave propagation algorithm.

Approach \ image sizes	1x1	5x5	10x10	20x20	40x40	80x80	160x160	320x320
Wave Propagation	0.0063	0.0081	0.0108	0.0144	0.0381	0.121	0.482	2.298
Ogniewicz's Approach (320x320)				0.382				

Table 7.2: This table shows the times (s) that are spent in the wave propagation algorithm and the Ogniewicz's technique for detecting symmetry set of the edge map in Figure 7.2. The quality of results in the wave propagation method do not depend on the size of image domain. The comparable results in terms of accuracy are obtained by the Ogniewicz technique for a image size greater than 320x320.

drastically whereas the number of discrete set of points required to trace the contour did not change for Ogniewicz's algorithm.

7.2.2 Limitations of Voronoi Diagrams of Discrete Objects

The symmetry representation of objects in images can be extracted from the Voronoi Diagram of these objects. However, observe that the symmetry sets extracted by these methods (or any method which reports accurate results!) are very sensitive to the quantization ef-

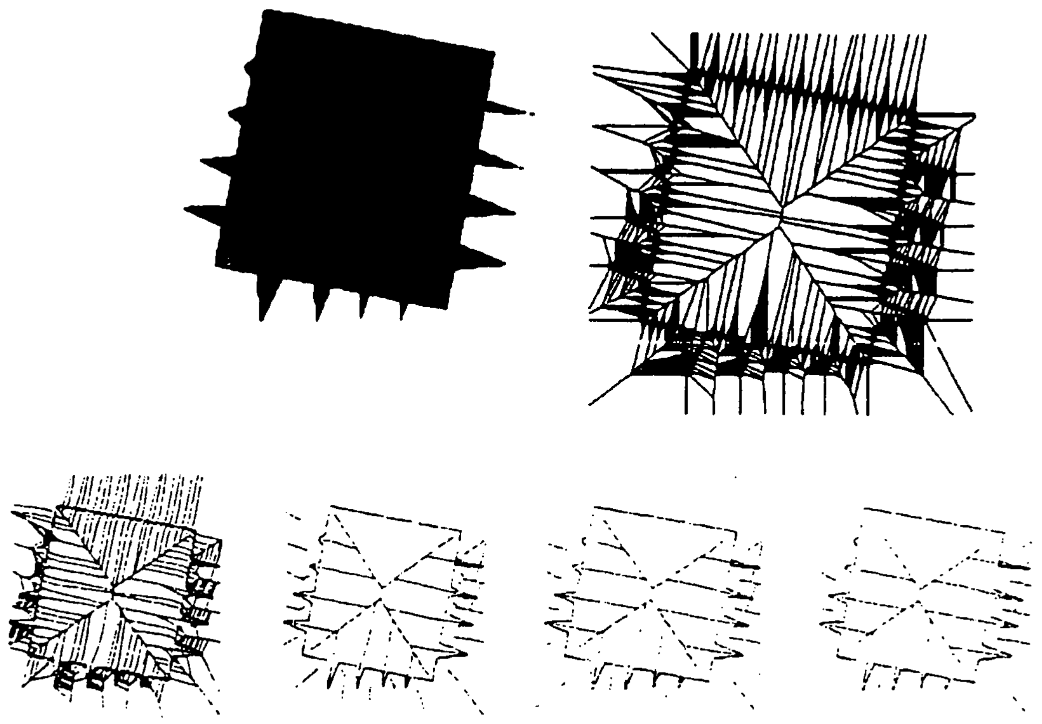
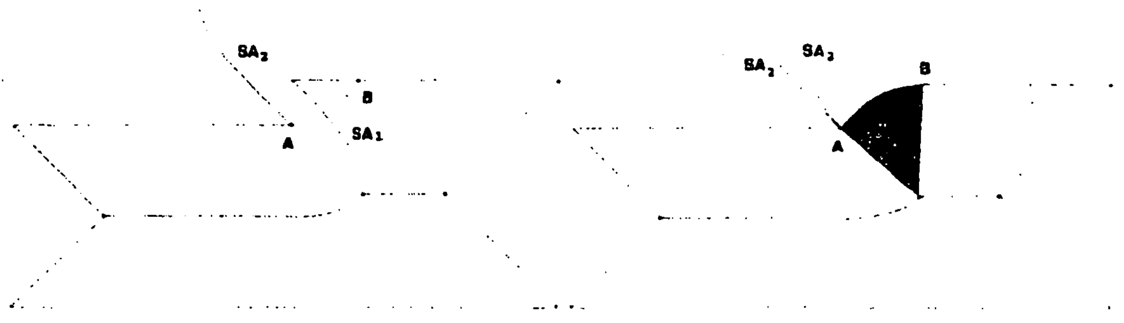


Figure 7.3: This figure illustrates pruning of the Voronoi Diagram of a binary image. (Top) Original binary image and its Voronoi Diagram representation. (Bottom) The Voronoi Diagram of the same shape after prunning branches whose salience is less than 1, 5, 10, 15. Note that higher threshold values result in the removal of important branches.

fects introduced by the discretization and geometric transformations [149]. A secondary pruning (regularization) process is needed to remove unnecessary symmetry axis branches. The most popular pruning processes are based on removing the symmetry axis points (or branches) whose saliency values are less than a certain threshold [149, 190]. Figure 7.3 illustrates an example of a prunning process proposed by Ogniewicz [149]. Observe that higher threshold values result in the removal of important axis branches, *e.g.*, corners of the square. In fact, it is not easy to distinguish important object features from insignificant ones at this stage of shape analysis. This is due to the fact that operations in the skeletal domain do not correspond to operation in the shape domain, as described below.

The regularization of symmetry branches correspond to the smoothing (or grouping in case of gaps) boundary segments. Let us consider the example shown in Figure 7.4 where the symmetry axis branch, SA_1 is to be removed. This requires that the boundary between A and B be modified so that no symmetry axes form in the shaded region. This can be



(b)

Figure 7.4: This figure illustrates the coupled symmetry/boundary smoothing algorithm to remove the discretization effects on a single step edge. First, the symmetry axis, SA_1 is deleted from the symmetry map. This requires that the boundary between A and B (the boundary foot points of the branch) be modified so that no shocks (or symmetries) form in the influence zone of boundary segment AB, (shaded region). This is accomplished by replacing the boundary segment with a smooth boundary segment, *e.g.*, circular arc. Observe that when the boundary segment AB is replaced, the symmetry axis SA_2 is no longer valid. Thus, the re-computation of symmetry axes and their saliency is axes required. $\overline{SA_2}$ is the correct symmetry branch. The VD approach requires significant computational expanse to implement this idea.

accomplished by replacing the boundary segment with a smooth boundary segment, *e.g.*, circular arc. Observe that when the boundary segment AB is smoothed by a circular arc, the symmetry axis SA_2 is no longer valid. Thus, the re-computation of symmetry axes and their saliency are required. Unfortunately, most pruning methods [149, 190] ignore the fact that removing an axis branch must result in smoothing a boundary segment and thus, requires a re-computation of the symmetries due to the replacement of the boundary segment with this smoothed boundary segment. However, this approach leads to two distinct problems for VD approach. First, the addition of a re-computation of skeleton and salience is prohibitively expensive for the VD methods. Second, smoothing implies the use of some boundary model, forcing a computation of VD for free-form curves, thus restricting the use of discrete Voronoi Diagram algorithms.

7.2.3 Voronoi Diagrams Of Curved Objects

Observe that image structure is rarely in binary form, and in fact the human visual system has developed the ability to detect features at resolutions an order of magnitude better than retinal resolution (hyperacuity). Many low-level vision algorithms have also developed

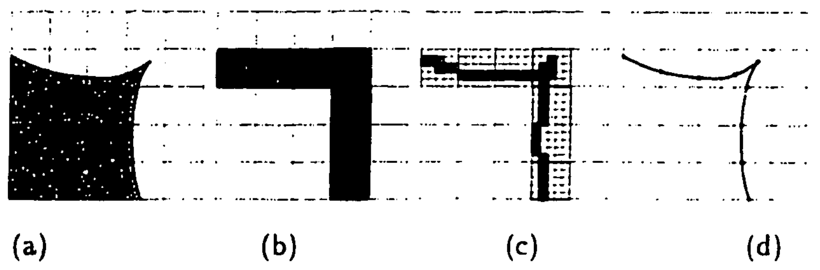


Figure 7.5: This figure illustrate that a geometric model is needed for accurate and efficient subpixel boundary representation. The binary representation of a curve (a) on a discrete grid (b) discards much of the geometric information. Higher resolution representation around the boundary (c) is also jagged and does not represent geometric information well. An alternative method to fine sampling is the use of models such as piecewise circular geometric model which maintains coarse grid sampling but allows for a representation of geometry within the pixel (d) [198].

similar capabilities [55, 202], which immediately raises challenging issues: how to represent each boundary with subpixel resolution *without* recruiting tremendous memory and computational resources? How to represent multiple curves within a pixel? How to represent singularities of curves? The intuitive approach of increasing the underlying grid resolution only at boundary points leads to a jagged representation, does not explicitly represent singularities, and imposes unnecessary memory and computational demands. The use of a piecewise circular geometric model of a curve, on the other hand, allows a model of curvature as well as a model of singularities, if needed [198].

Only very recently, the Voronoi diagram of curved objects has been studied [237, 3, 44, 43, 166, 167]. One of the main difficulty in computing Voronoi Diagram of free-form curves stems from the need for the accurate and robust detection of the bisector between these curves [84, 65, 64, 60]. A second difficulty is how to integrate these bisector that underly the construction of the Voronoi Diagram.

We now present the two approaches to computing Voronoi diagrams of curved objects since they are interesting and closely related to the work presented in this paper.

First, Chou [44] presented an interesting algorithm for computing the VD of piecewise circular shapes by tracing each VD branch using the differential properties of the diagram and the boundary up to the places where it does not satisfy the symmetry set requirements. Whenever two branches collide a new branch is initiated. However, (*i*) this algorithm is applicable only to closed shapes, so that it is not applicable to curve segments, *e.g.*, frequent in edge maps, (*ii*) it is not computationally efficient due to the tracing of each branch until it ends, and (*iii*) topological changes are not detected.

Second, Ramamurthy and Farouki [166, 167] have developed an excellent algorithm for detecting Voronoi diagrams and medial axis transform of planar free-form curves. Their algorithm is based on incremental bisector computation. Specifically, the algorithm starts with a single boundary element and constructs its Voronoi Diagram. In successive iterations, a new boundary element is added to the boundary list and the Voronoi Diagram of this list is updated, Figure 7.7 as follows. Suppose that the Voronoi Diagram of m ($m < N$) boundary segments is constructed at m step. At step $m+1$. (*i*) the bisector of the new boundary segment, b_{m+1} with each of the previous boundary segments, b_i for $i = 1, ..., m$ is computed, *i.e.*, m new bisector 7.7 (left column); (*ii*) Voronoi regions of the each boundary segment, b_i are partitioned by these new bisectors; (*iii*) the Voronoi region of the new segment is constructed as the union of regions obtained at (*ii*) Figure 7.7 (right column). The core of the algorithm is the robust and efficient computation of the bisector between two free-form curves. It is well-known that the bisector of linear/circular boundaries can be computed *exactly*. However, the bisector of conic, cubic or higher-order boundaries must be approximated. In fact, Farouki and Johnstone [65, 64] studied the computation of the point/curve and curve/curve bisectors. Specifically, in [65], they compute the *untrimmed* bisector between a point and a parametric regular curve by using the variable distance offset curves. The true bisector is computed by trimming the segments that do not satisfy the minimum distance criteria, Figure 7.6. The curve/curve bisector can be computed as the envelopes of the point/curve bisectors [65, 64].

Unfortunately, the Voronoi diagram algorithm of Ramamurthy and Farouki [166] is computationally expensive. Specifically, it requires $\binom{N}{2} = \frac{N(N-1)}{2}$ bisector computations for the set of N boundary model. It should be noted that curve/curve bisector computation is computationally expensive since no priori information is available about the true bisector of two boundary segments. Thus, much computation is wasted for the computation of those bisector segments that are not part of the true Voronoi Diagram, *i.e.*, they will be pruned at some point in the incremental algorithm, Figures 7.7 and 7.8. In addition, the Voronoi regions of boundary segments are modified several times (at most N times) and in fact, these modifications require computationally expensive and error-sensitive "bifurcation" point (junction points in our shock representation) computation. Furthermore, the numerical errors in incremental Voronoi Diagram algorithms summarized in [207] could become significant if N is a large number. Specifically, the errors in the computation of

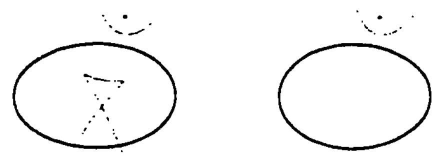
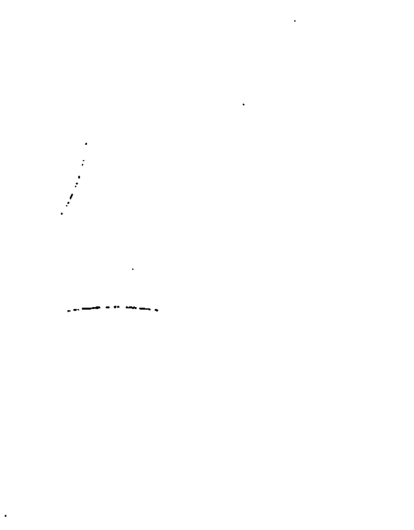


Figure 7.6: (a) The untrimmed bisector between a point and an ellipse. (b) The true bisector after a trimming process.

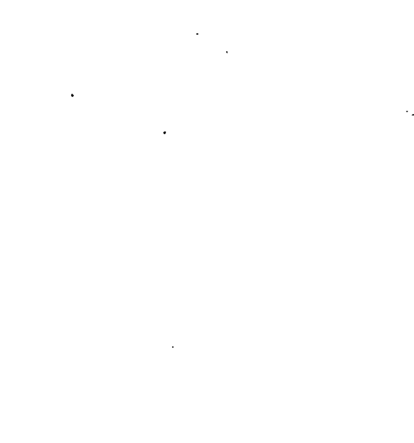
Voronoi regions in each step of the algorithm can accumulate and become significant.

We believe that our wave propagation algorithm provides a remedy to the computational issues of the Voronoi diagram algorithm of Ramamurthy and Farouki [166]. Specifically, (*i*) *only* the true bisectors of two boundary models are computed at every step of our algorithm. thus eliminating the trimming process. Figure 5.3 and Figures 5.3 and 7.9a illustrate the computation of the "true" bisector between a line and circular arc segment as a alternative to Figure 7.8. (*ii*) *only* the valid boundaries of Voronoi Diagram are computed. Figure 7.9 illustrates the computation of the Voronoi Diagram of five boundary elements. Unlike the construction of the Voronoi Diagram by the incremental approach of Ramamurthy and Farouki shown in Figure 7.7. *only* valid boundary segments the Voronoi Diagram are constructed and the Voronoi regions of each boundary model is never modified. (*iii*) Our method is less sensitive to numerical errors due to the simultaneous representation and propagation of wavefronts and shocks which keep zones of influence separate and lead to exact results. For example, in the incremental Voronoi diagrams the computation of bifurcation (junction) points are very sensitive to the numerical errors whereas in our approach we can compute exactly whether two bisectors intersect with each other during the wave propagation, thus reducing the errors in Voronoi diagrams caused by numerical errors.



(a)

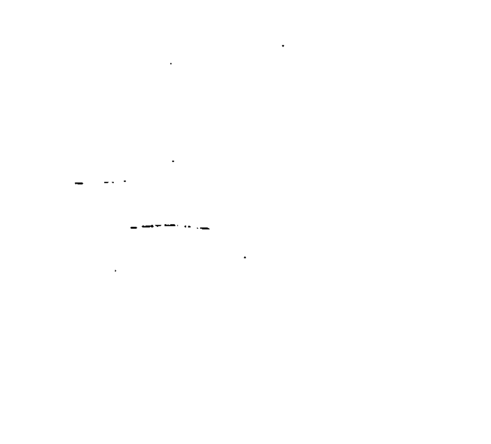
(b)



(c)

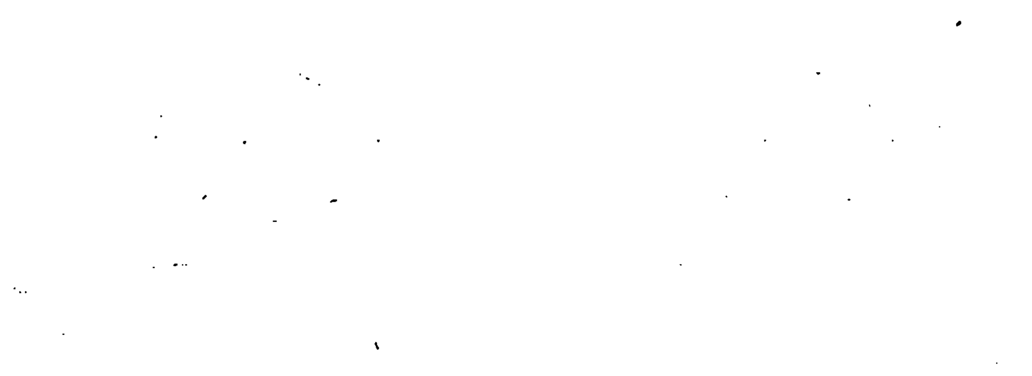
(d)

(e)



(f)

$\Rightarrow$



(g) (h)

Figure 7.7: This figure illustrates computation of the Voronoi diagram of five (5) geometric boundary models by an incremental approach. At each step bisector curves between a new boundary element and each of previous boundary elements are computed are drawn in red. The true bisectors after trimming process is shown in blue while the removed branches are shown in grey. Specifically, left column in each step corresponds to the addition of a new boundary element to the Voronoi Diagram of the previous boundary elements. Right column in each step corresponds to the modification of Voronoi Diagrams. This algorithm is rather inefficient because many bisector which may not be play a role are nevertheless computed. The algorithm is of order $O(N^2)$.

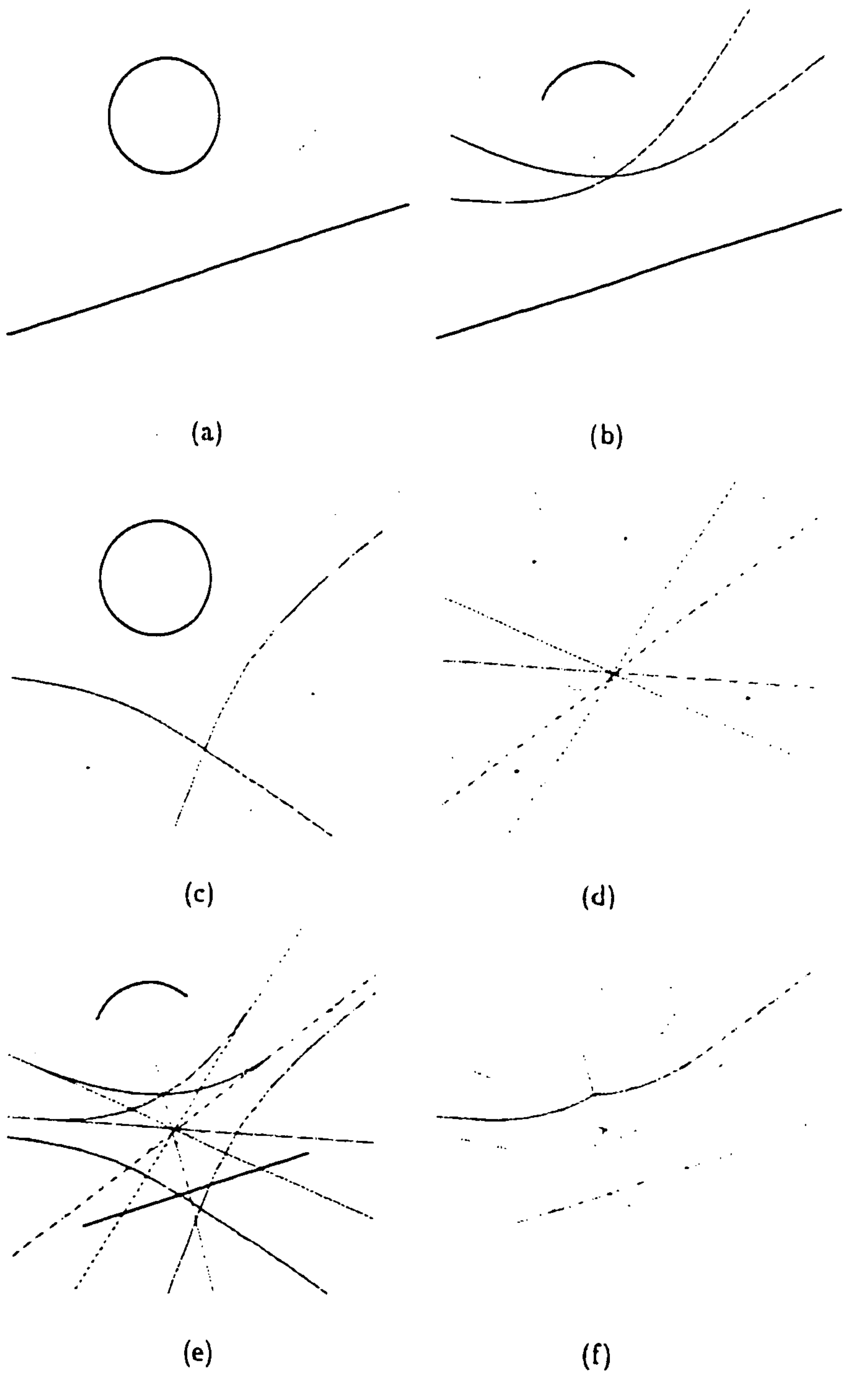
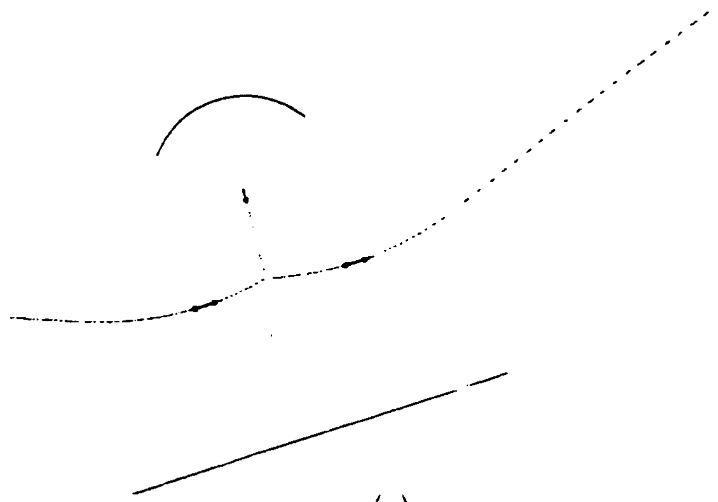


Figure 7.8: This figure illustrates the computation of the true bisector between a circular arc segment and a line segment. (a) the bisector curve between the *full* circular arc and a line; (b) the bisector curves between the line and the two end points of the circular arc segment; (c) the bisector curves between the circular arc and the two end points of the line segment; (d) the bisector curves from the end points of the circular arc and line segment; (e) the untrimmed bisector curves between the circular arc and line segment. (f) the true bisector (blue) is detected by a trimming process.


(a)

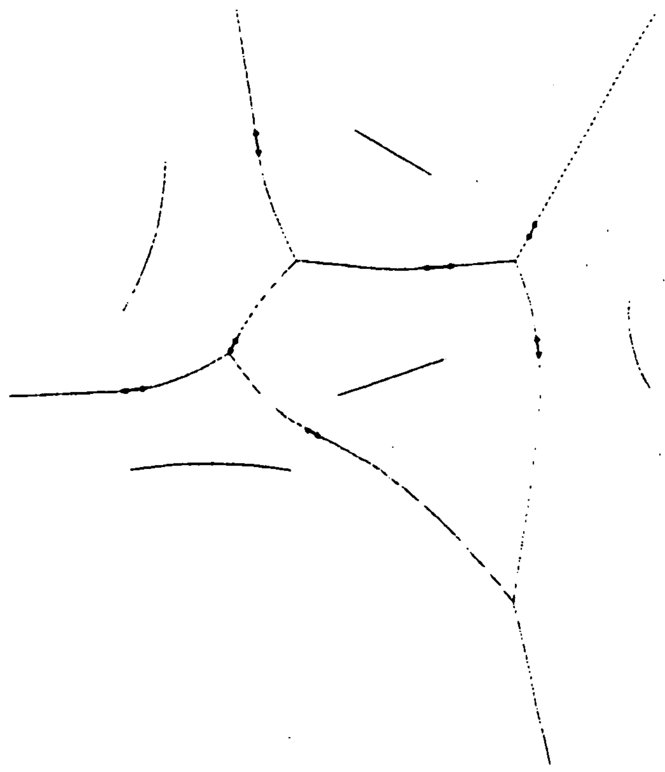

(b)

Figure 7.9: (a) This figure illustrates the detection of true bisector curves between a circular arc segment and a line segment by the wave propagation algorithm. (b) This figure illustrates the computation of the Voronoi diagram of five (5) geometric boundary models by the wave propagation algorithm. This method in combining wave propagation with bisector computation is rather efficient because only the bisectors which are needed are computed. The shockwaves in this example form from the second-order shocks which then initiate two shockwaves moving in opposite directions (black vectors).

Part II

Perceptual Organization Via Symmetry Maps and Symmetry Transforms

Chapter 8

Symmetry Transforms

We now propose that the symmetry map of the edge map of an image serve as an intermediate representation mediating between low-level edge maps and high-level object boundaries. We present a set of transformations acting on the symmetry map and which produce symmetry maps that in the edge map domain correspond to smoothing noisy edge segments, grouping broken edges, detecting the visual parts of objects, and removing the affects of occlusion. The symmetry map fully represents the initial edge map and any contour map resulting from grouping the edge elements [74]. Thus, each visual transformation has a well-defined counterpart as a transformation on the symmetry map, called *symmetry transforms*. Symmetry transforms include *splice*, *gap/part*, *loop* and *non-blending occlusion* and operate on the symmetry set and object boundaries. Specifically, splice transform removes a branch of the symmetry map, modifies related symmetries and smooths boundary in this process. The gap/part transform detects a visual part or gap and replaces it with a *part-line* by connecting the related symmetries. The loop transform detects spurious edge elements and corrects the distorted symmetries that are altered due to their presence. The (non-blending) occlusion transform detects occlusions on the edge map and corrects symmetry map as well as boundary.

It should be emphasized that these transformations are *local* operations on the symmetry maps and are very effectively implemented by the explicit wave propagation algorithm earlier in Chapter 5. Specifically, we define two canonical shock operators which these symmetry transforms are based on.

It has been observed that these symmetry transformations cannot be applied independently. Figure 8.1 illustrates an example of a noisy curve segment with a part. Siddiqi and

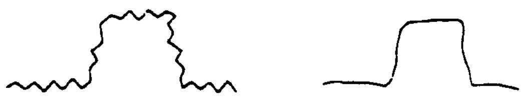


Figure 8.1: The parts of objects are often distorted by noise. Their robust detection requires that variations on the boundary be removed without removing globally salient features. *e.g.*. corners. (a) A noisy curve segment with a part. (b) The outline of this curve is smoothed by a splice symmetry transform, Chapter 9.

Kimia [195] presented a limb-based partitioning scheme in which first candidate part-lines that pass through pairs of negative curvature minima are detected and the most salient ones are sequentially selected. This approach approach works well in most cases. but cannot detect the part in Figure 8.1 due to the presence of noise on the boundary which produces many curvature minima and thus disturbs tangent and curvature measurements at coarse scale. The smoothing of boundary segments to remove unwanted noise also results in removal of the globally salient small scale structures. Similar problems affect the grouping of edge segments.

The important remaining question (which is not a focus of this thesis) is that given the option of applying a number of symmetry transform which should be applied? Similarity of two objects in the shape from deformation approach of Kimia [99] is viewed as the magnitude or extent of the deformation needed to bring one into register onto another. There. the search for the notion of similarity was based on the intuition that slightly deformed objects often appear similar. A traditional difficulty with the notion of similarity as the cost of the deformation path is the high dimensionality of the space of deformations. In [225] each deformation has been reduced to a set of shape transitions motivated by a shape transitions, thus significantly reducing the dimensionality. These transitions represent the shape deformations where the symmetry map itself undergoes an abrupt change, with a slight change in the shape, as opposed to the typical case where it changes continuously. Each transition then corresponds to a symmetry transforms. These transformations punctuate global deformation paths and segment them into portions where the representation changes continuously with shape deformation, Figure 8.2. The task of object recognition is then

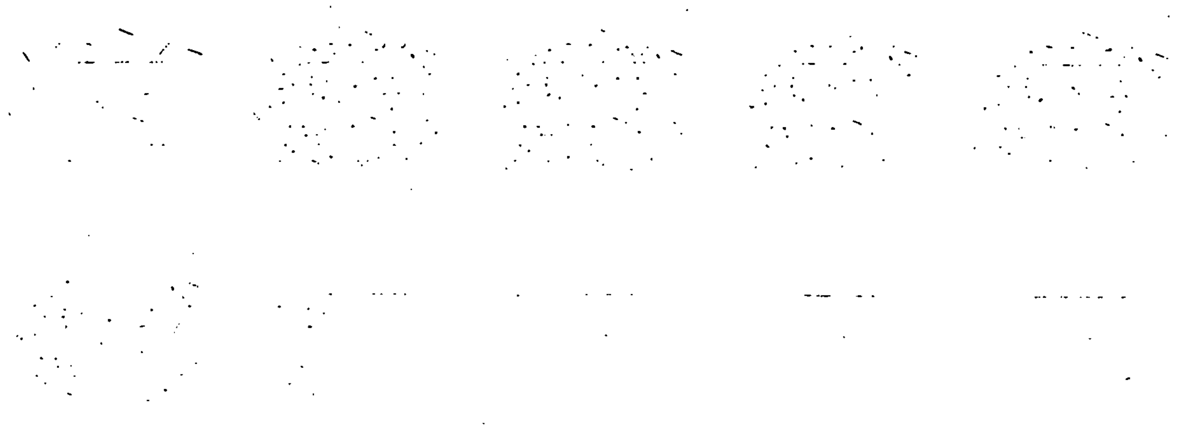


Figure 8.2: The figure illustrates a particular sequence of *symmetry transforms* (loop and gap/part) that leads to the recovery a complete rectangle. Of course, other sequences can be applied, but they do not necessarily lead to complete contours or forms.

that of picking the sequence of transitions which characterize the least action path, Figure 8.3. It should be noted that the contribution and focus of this chapter is the proposal that a symmetry map be used explicitly as an inter-meditate level representation between low-level edge maps and to show that all grouping operations are naturally expressible in this language. The task of selecting the optimal sequence is addressed in the edit distance approach [225].

8.1 Transitions of Symmetry Sets of Closed Shapes

The rich variety of visual transformations deform the shape of objects in many dimensions. Maintaining the sense of an object in the course of these deformations requires maintaining the stability of the underlying shape representation with these variations. Unfortunately, skeletal or symmetry-based representations sometime experience numerous abrupt changes, or *transitions* with slight changes in the shape but this happens only for select (non-generic) shapes. A classical example of such instability is that of a small protrusion in the shape which gives rise to a large branch of medial axis, Figure 8.4 (top). Similarly, slight change in the location of one of branches, Figure 8.4 (bottom) introduces a second type of instability [75].

Giblin and Kimia [75] classified the transition of the medial axes and subsequently the corresponding transitions of the shock structure. Specifically, six types of shock transitions

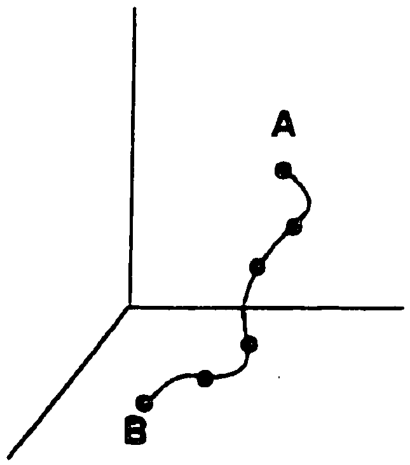


Figure 8.3: A shape A undergoing an arbitrary sequence of deformations to transform to shape B.

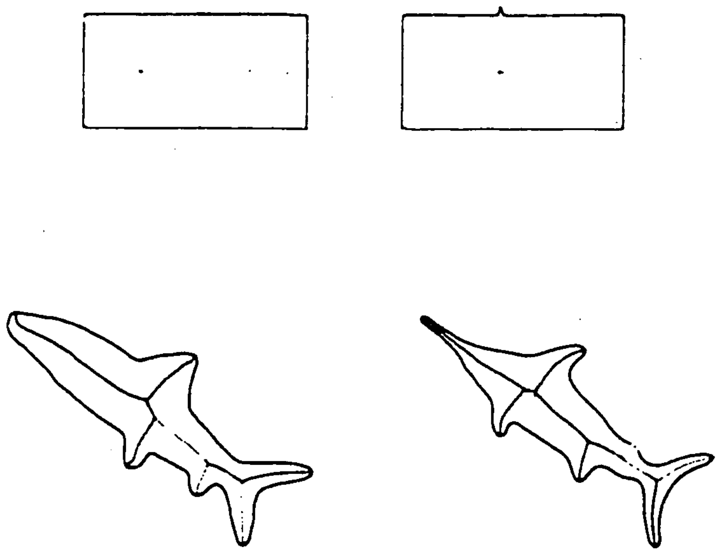


Figure 8.4: Two classic instabilities of the medial axis: (top) a small protrusion gives rise to an unproportionately large branch; (bottom) a small shift in the location of the lower fin of the fish drastically alters the graph structure of the skelet n, thereby causing difficulties for graph matching algorithms for object recognition.

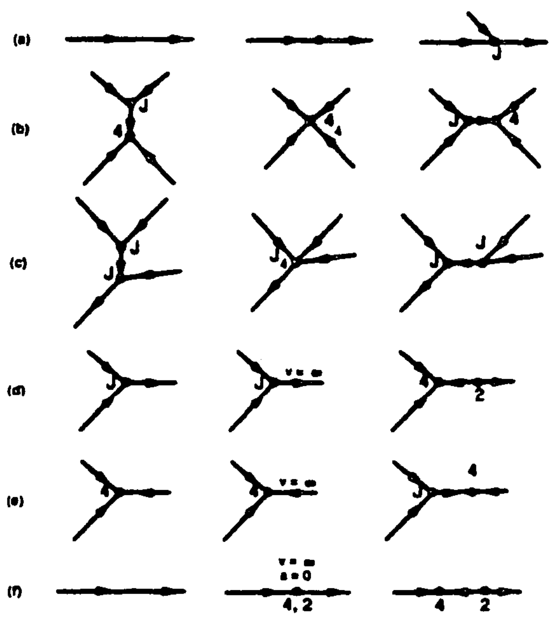


Figure 8.5: Transitions of the shock structure. *(a)* is the A_1A_3 transition. *(b)* and *(c)* are the two types of A_1^4. *(d)* and *(e)* are the two types of A_1^3 with infinite velocity, and *(f)* is the A_1^2 point with infinite velocity and zero acceleration. The operations to make the right and left columns equivalent are: splice, contract (two types), and merge (three types).

are identified *for closed shapes*, Figure 8.5. A shock transform is then required to make the equivalence class of shapes on either side of the transition adjacent, hence "seaming up" the shape space and allowing for a regularization of the shock structure. These in turn lead to the following transforms[1] defined in [225]:

Definition 3 Splice Transform: *remove branch at A_1A_3 points and modify related shocks accordingly (transition a).*

Definition 4 Contract Transforms: *bring together closeby A_1A_3 points (transition b and c).*

Definition 5 Merge Transform: *bring together closeby A_1A_3 points and infinite velocity A_1^2s (transition d, e, and f).*

In general, these transitions are motivated from a symmetry-based recognition framework. Specifically, the practical use of symmetry representation often relies on matching

[1]The notion A_k^n implies a point with a circle tangent with contact of order k at n distinct points.

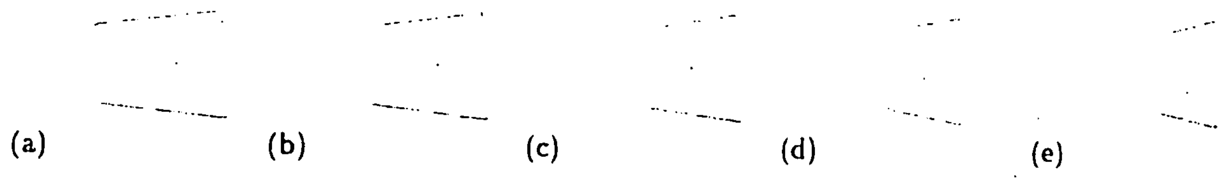


Figure 8.6: In the process of compressing a shape (left to right) the shock graph experiences a transition (d) where the pairing of branches is reversed, making mateling of the extreme left and right structure difficult. We follow an edit distance approach with explicit edits to make the shapes on the either sides of the transition (b) and (e) by explicit transformations.

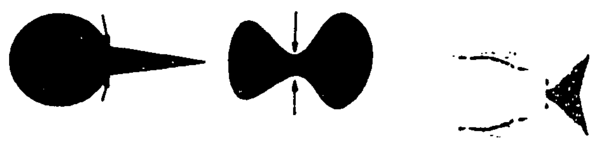


Figure 8.7: Two types of parts: limbs (left) and necks (middle). The limbs and necks of the fish shape (right) are shown [195].

the qualitative structure of the shape. which is represented as a graph. The six instabilities depicted in Figure 8.5 occur frequently, thus altering the qualitative structure. The symmetry transformations can then be performed on a shock graph to alter them in such a way as to simplify the represented shape. Tirthapura *et al.* [225] proposed an approach for measuring distances between shock graphs as the cost of the "least action" path consisting of sequences of elementary symmetry, (or *edit* operations. The main idea is to measure the distance between two shapes A and B as the minimum cost of deforming shape A to shape B, Figure 8.3 and Figure 8.6

In the following sections, we augments the closed shape transitions with those of open curve segments of an edge map, and propose three additional symmetry map transforms for them. The first type of transforms namely, part and gap, is motivated by the deletion of a curve segment. The second type of transforms, loop and (non-blending) occlusion transform, arises from the addition of a boundary element to the edge map.

8.2 Part/Gap Transform

In the real world, we often do not see entire objects due to gaps, occlusion, *etc.*, but we are still able to recognize them without any special effort. In general, objects are often perceived

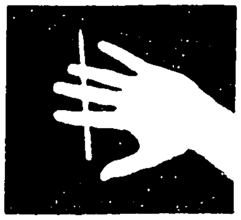


Figure 8.8: This figure depicts a case where an occluder blends with the occluded object.

as a hierarchical structure of simpler elements [145, 200]. Thus, organization of objects and their parts into classes and hierarchies is an important step in object recognition. For example, object recognition based on the global structures faces major difficulties, especially in the presence of occlusion, whereas the approaches based on the local description of parts are less affected. In fact, stable detection of a few parts is often sufficient for a robust recognition.

A theory for partitioning visual form was proposed in [195] leading to two types of parts: one at *necks*, or the narrowest regions, namely, at part-lines whose lengths are a local minimum, *and* which are the diameter of an inscribed circle. (These necks correspond to the second-order shocks of [112]), and *limbs*, or curves through two negative curvature minima that smoothly interpolate the tangent at one point to the tangent at another. Figure 8.7. The computation of each candidate part-line involves computing a measure of salience which is then used to resolve conflicts when they arise [195]. While this framework is developed for partitioning *segmented* shape, the computations are *local* with respect to the part-line size and can thus be computed *prior* to figure-ground segmentation if appropriate contours are identified.

In this thesis, we also believe that parts are important in shape representation because they represent significant and discrete changes in their object representation. Recall that our final goal is to construct a shape space for object recognition based on symmetry transforms, which capture the discrete changes in this space, Figure 8.3. Therefore, we propose a reformulation of limb and neck-based partitioning approach in the shock language using shock labels, which allows parts to be computed directly from images.

Let us first try to understand how the parts of objects are formed in nature and how they appear in two dimensional images. Consider, for example, the branches of a tree which can be thought of the parts of the tree. The formation of a new branch (or leaf) can be seen as a discrete event. Once a branch forms, its growth may be characterized

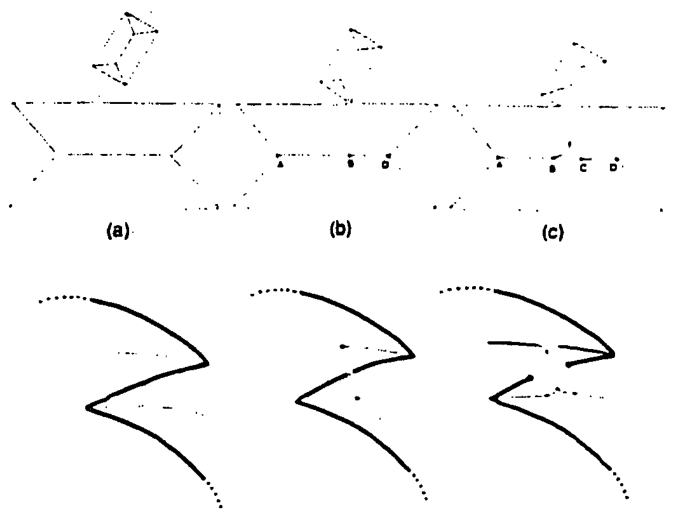


Figure 8.9: This figure depicts the intimate connection between parts and gaps. (TOP) The formation of a part from an occluding object which blends with the occluded object. Observe that the initial point of contact creates a degenerate shock identical to the creation of a gap. (BOTTOM) The gap shock transition. As a point is removed from the boundary, a degenerate branch and two new A_1^3 (junctions) are added to the symmetry map. As the size of gap is increased, this gives birth to a a pair of semi-degenerate or degenerate branches from each junctions.

by a continuous process until a new branch forms from it: Leyton presented a view of shape [126]. In man made objects, parts are often glued to another (bigger) object by some special technique. The photometric projection of the 3D world into to 2D images, however, may present more parts than what is present in the real world. For example, some visual parts may be erroneously created by the photometric visual transformations, *e.g.* occlusion, or by low-level vision algorithms, Figure 8.8.

How does the formation of a part modify the symmetry representation? Let us consider a formation of a part from an occluding object which blends with the occluded object, Figure 8.9a. Observe that the initial point of contact creates an infinitesimal gap in the contour. This deletion of a contour point gives rise to a degenerate shock branch orthogonal to the contour and two junctions. Further occlusion results in the enlargement of the gap which then causes the formation of two semi-degenerate branches in the shock representation of the occluded object. Conversely, a junction with two semi-degenerate shock branches implies the existence of two end points, which in turn signals the possibility of a part.

Observe that the introduction of a part leads to a gap on the boundary curve. Gaps on edge maps or object boundaries due to visual transformations, *e.g.*, shading, occlusion, or due to the limitations of edge detectors follow an identical transition, Figure 8.9 (bottom).

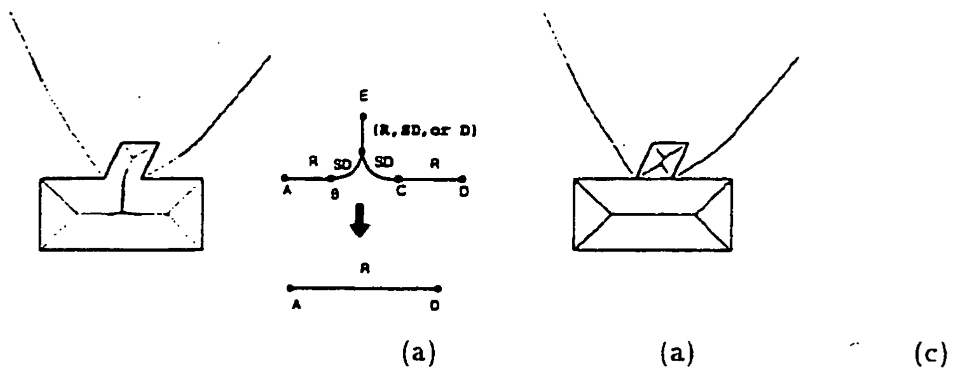


Figure 8.10: Enforcing skeletal continuity across these flanking branches leads to partitioning the part. (a) computed shocks, (b) part-transform, (b) and its application to the shocks computed in (a).

Specifically, as a point is removed from the boundary, a degenerate branch is added to the symmetry set. The end points of this new branch are the two junctions that connect the degenerate branch to existing branches. The removal of a finite segment (enlarging the gap) leads to the birth and growth of two semi-degenerate (or degenerate) branches from each junction. Thus, the gap and part transitions emphasize the intimate connection between parts and gaps (null parts), which was previously illustrated in [201].

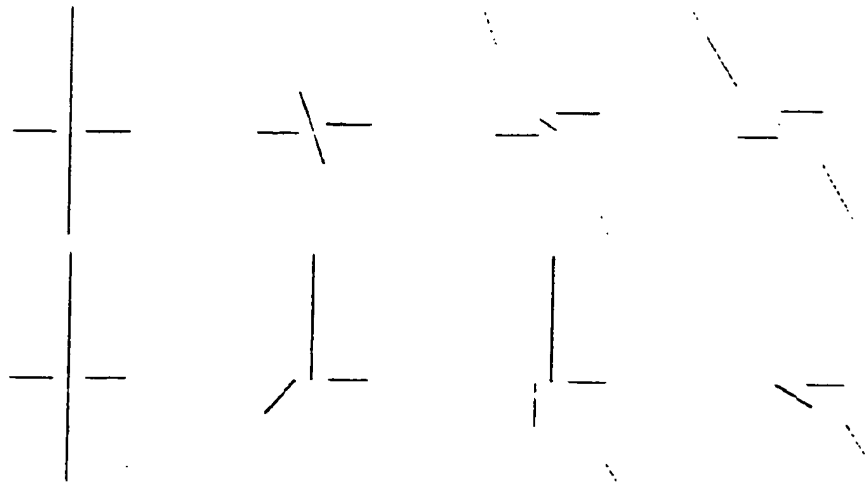
In both transitions, observe that the continuity of the skeletal piece is violated by the introduction of the part or the gap. *Shape continuity* implies skeletal regularization by removing the semi-degenerate portion from B to C and smoothly continuing AB to CD. Figure 8.10. This in-effect enforces the completion of the gap. This is accomplished by sending waves from the completion contour at points of degeneracy Figure 8.10.

Definition 6 The gap/part transform *acts on a shock path between a pair of nodes in the shock structure, each with a (semi-)degenerate branch, replaces this path with the shock structure resulting from waves emanating from the completion contour at point sources of degeneracy. Note that the gap/part transform results in a segregation of the connected symmetry graph at this point.*

8.2.1 Role of Saliency In Part/Gap Transforms

Salience of a limb is partially related to the optimal shape of the completion contour, or the limb part-line [195]. Figure 8.11 examines the relationship between salience and shock labels: observe how the length of the degenerate shock group corresponding to the gap

(b)



(d)

Figure 8.11: The length of degenerate shocks signals the saliency of the part-line/completion contour. Note that semi-degenerate shocks are shown in yellow.

varies with the "extent' of the part. and the " misalignment" of two tangents. Figure 8.11.

Salience of Figure 8.12 illustrates how partitioning and grouping are identical operations in the shock domain. Figure 8.12a is perceived as two overlapping triangles. As parts are removed one by one. Figures 8.12b and 8.12c. the first triangle is recovered. resulting in two disjoint pieces of the second triangle. Subsequent removal of parts (background) leads to the grouping of two disjoint pieces. Waves from the pair of limb part-lines. previously referred to as "hidden limbs" [195], now form the shocks corresponding to the second triangle. Observe that the notion of salience is crucial when the alignment and smoothness of the connecting limb part-line is challenged. Figure 8.12f.

August *et al.*, [15] presented an interesting connection between certain portions of a skeleton, namely the "gap skeleton" and the skeleton of a virtual occluder which signals the grouping of contour fragment endpoints. The main idea is that occlusion leads to T-junction discontinuities. They propose that it is not the shape of the optimal continuation curve at T-junctions that is significant, but rather the consistency of the grouping as the outcome of some virtual occluder. Thus, based on the assumption that extracted contours have been partially depth-ordered in separate maps based on T-junctions, they group end points with no contour present in the circular neighborhood whose diameter is the line segment joining the two endpoints. Expressed in the language of shocks, this implies a second-order shock and two first-order shock groups (gap skeleton) which arise from the end points.

Our approach bears some resemblance to this proposal where the main similarity is the expression of contour grouping in the language of shocks, specifically that certain degenerate

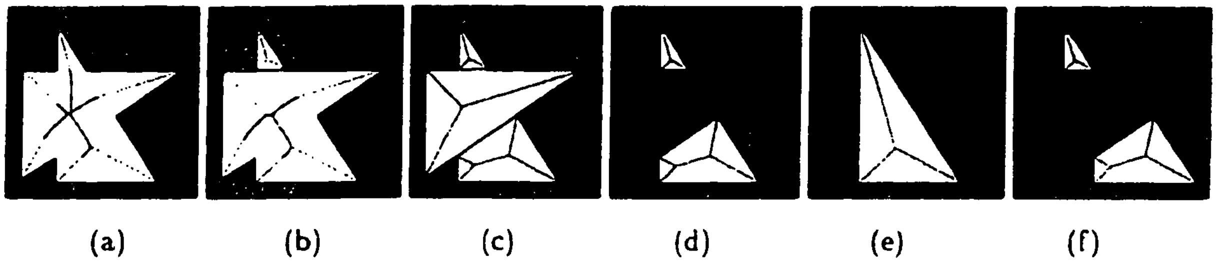


Figure 8.12: (a-e) partitioning and grouping as identical operations lead to the recovery of two triangles. (f) the role of salience of part-lines in grouping when alignment is challenged.

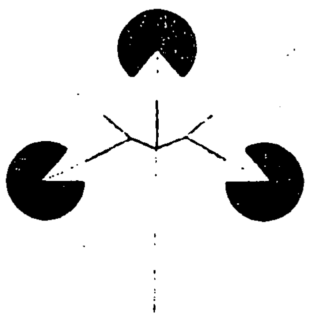
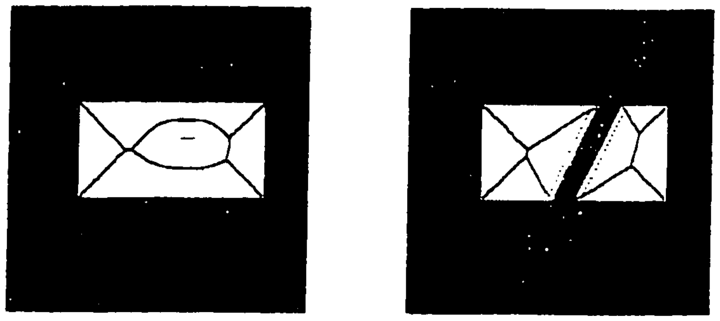


Figure 8.13: Degenerate shock sets signal contour grouping, some indicate the completion of triangle, while others indicate the completion of the pacman figures. First, observe that separation in depth which is assumed in [15] is not required here but rather follows from shock labeling and subsequent contour grouping. Second, this example illustrates that rarefaction waves generated by corner points are also significant

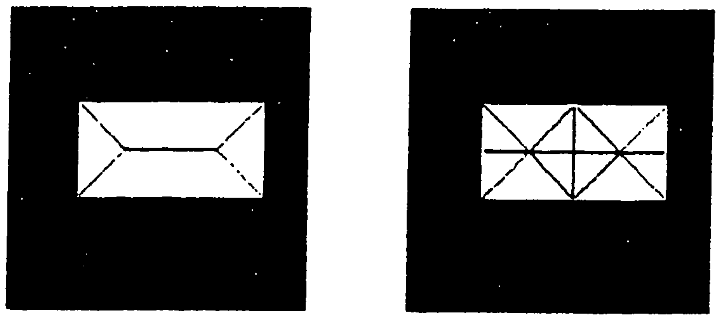
first-order shock groups, arising from end points (not corners) *and* containing second-order shocks constitute the gap skeleton. However, there are a number of significant distinctions. First, the motivation and domain of application is rather different: in contrast to grouping contour fragments partially ordered in depth via T-junctions, our goal is to extract partial shocks from partial contours of the *full set of extracted contours* by separating extracted symmetries into classes: regular shocks represent true contour symmetry, while degenerate shocks represent connectivity. Thus, this approach achieves such a grouping without depth segmentation via limb hypothesis and inter-penetrating waves which in turn *lead to* depth segregation, Figure 8.13. August *et al.* require a separation of the triangular contours from the circular ones to achieve grouping, while we propose that a separation follows from the grouping.

This figure also illustrates a second important distinction, namely, the concept of classi-



(a) (b)

Figure 8.14: Symmetries are often distorted by (a) spurious edges and (b) occlusion.



(a) (b)

Figure 8.15: (a) Symmetry axis transform and (b) full symmetry of a rectangle.

fication of a wave based on whether the orientation of its front can be reliably related to the original front (regular) or not (rarefaction wave). Thus, corner points which generate rarefaction waves lead to degenerate shock groups which are not gap skeletons. Third, we hold that the shape of the optimal contour connecting the end points is significant, indicating the salience of the grouping, *e.g.*, in detecting alignments, Figure 8.12 and Figure 8.11.

8.3 Loop Transform

Symmetry maps are also altered by spurious edges which arise from surface markings, texture edges, specularity highlights, noise, *etc.*, and occlusions such that the resulting symmetries are not recognizable, nor lead to figure/ground segmentation. In this section, we consider spurious edges which are special cases of non-blending occlusions where the occluder and the figure do not blend in. These two types of symmetry map distortions are illustrated in Figure 8.14. The non-blending occlusion will be studied in the next section.

The distortions in symmetry transform (or specifically symmetry axis transform) have

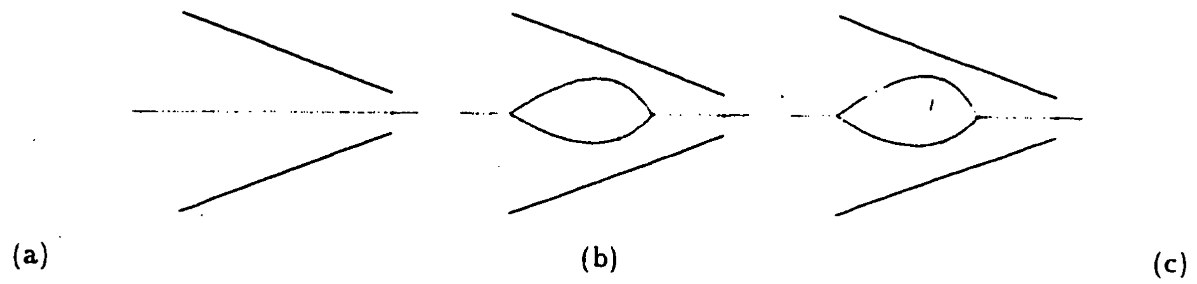


Figure 8.17: This figure illustrates that the effect of adding a single point to the edge map is drastic. (a) and (b). A loop is generated as a result of interacting with the surrounding elements. As the point grows to a finite size, the loop is deformed accordingly, (c).

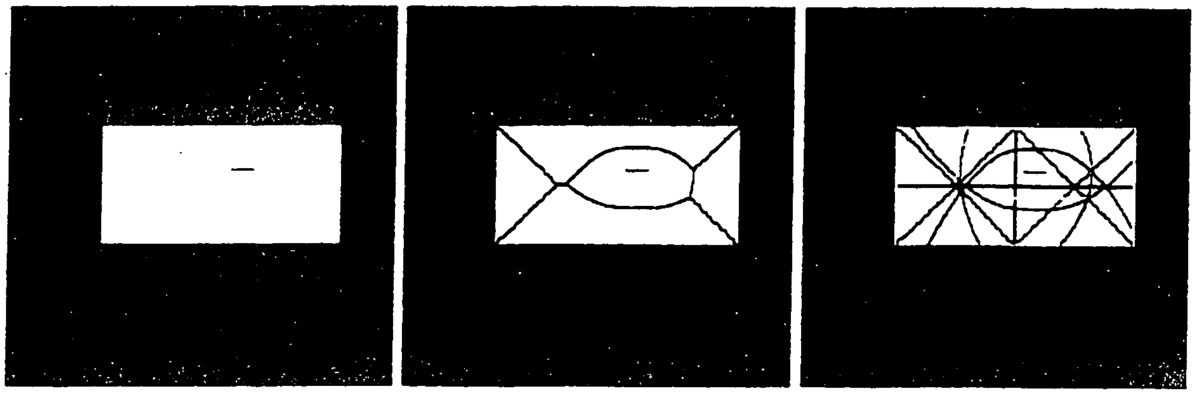
Proposition: The union of all generations of shocks is the full symmetry set.

Proof: Each point in the full symmetry set can be viewed as the quench point of two waves traveling without interruption from two boundary segments. Since wave initialized at quench points are the continuation of waves quenched at these points all two boundary segments eventually interact. Conversely, each multiple generation shock is clearly a point of the full symmetry set by the same argument.

The shock-based representation can implement such a process since complete information about the incoming waves is stored as shock location, time of formation and velocity. The second observation is that multiple generation waves and shocks can recover the distorted or missing symmetries. The idea is to launch second and further generation of waves only at *select groups of shocks* as indicated by special properties of the shock itself. For example, an isolated spurious edge or equivalently a hole in the object interferes with the formation of appropriate symmetries, Figure 8.17, but also always leads to the formation of a *loop* in the shocks.

The addition of a single point, an infinitesimal change in the edge map, causes a large change in the symmetry map. It creates second order shocks and pairs of (semi-)degenerate branches emanating from each and forming a loop which represents the quench locus of the wave fronts emanating from the added point, or its influence zone. The shock transform that regularizes this instability *i.e.* makes the edge maps on the either side of the transition (in Figure 8.20a and Figure 8.20c) equivalent is the *loop transform.*

Definition 8 Loop transform *acts on closed minimal loops of the shock structure and replaces them by the shocks arising from the interaction of waves propagated from the outer*



(a) (b) (c)

Figure 8.16: (a) Rectangle with spurious edge, and (b) skeleton of rectangle with spurious edge element. (c) full symmetry set: relevant symmetries are now embedded in a large set of symmetries.

prompted several approaches to extract the *full symmetry set* from unsegmented gray-scale images, Figure 8.15. Scott *et al.* [176] propagate waves to recover the full symmetries. They also suggest a convolution approach for implementing the full symmetry set. Pizer *et al.* [159] use a similar approach where by a voting scheme, edges measured at each scale vote for medialness at a point which is a constant proportion of scale away. The ridges of the resulting surface constitute the *core*, a skeleton in x, y and σ (scale). Kelly and Levine [96] use annular symmetry operators in a similar fashion to derive the full symmetry set.

It has been argued that since the full symmetry set represents all the symmetries of a shape, spurious elements and gaps affect the full symmetry set less than the SAT. We argue that while some of the full symmetry set is revealing the underlying object regardless of the transformation applied (addition of spurious elements), not all of it is useful, in that some additional unintuitive branches can in-fact lead to ambiguities for object recognition, Figure 8.16.

The sensitivity of SAT and the ambiguity of the full symmetry set prompts us to propose an alternative which is based on a view of the full symmetry set as the union of quench points of a series of waves:

Definition 7 *The* first generation wave *is the wavefront initiated at the edge map of an image. The* first generation shock set *is the set of quenched points (shocks) arising from the propagation of first generation wave. The* n^{th} generation wave *is the wavefront initialized at the* $(n-1)^{th}$ *generation shocks at the time indicated by its formation. The* n^{th} generation shocks *are the shocks corresponding to the* n^{th} *generation wavefront.*

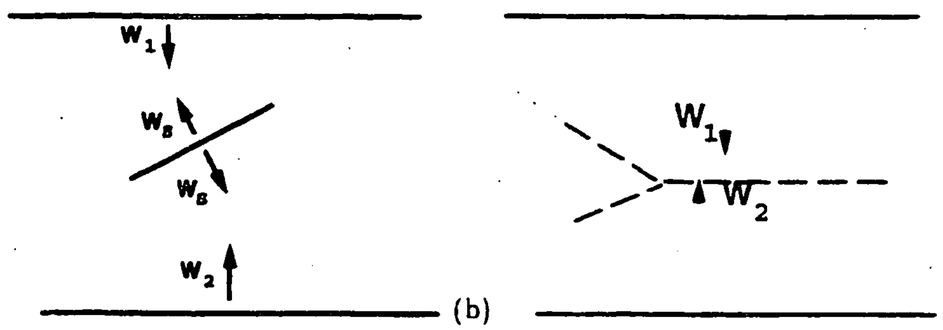
(a)  (b)

Figure 8.18: (a) waves W_1 and W_2, are quenched by waves from the spurious boundary, W_s, resulting in a loop in the shock structure. The shocks on the loop can now simulate the passage of the original flow of waves W_1 and W_2 via secondary waves, (b), which are initiated at the shock in a delayed manner, resulting in the formation of shocks due to the top and the bottom boundaries.

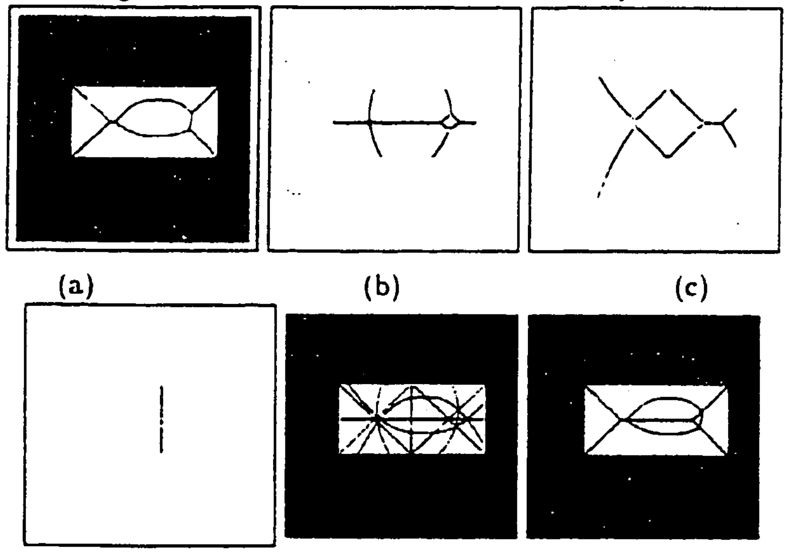

(a) (b) (c)
(d) (e) (f)

Figure 8.19: Multiple generation shocks of the image; (a), (b), (c), and (d) depict first-, second-, third-, and fourth-generation shocks, respectively. (e) the superposition of all generations of shocks constitutes the full symmetry set. Observe that un-intuitive nature of the full symmetry set; (f) second generation of waves exclusively initiated at the loops gives rise to shocks which complete the rectangular symmetry set.

region of the loop.

The loop transform allows wavefronts on the outer part of the loop to propagate to the inside of the loop, in effect removing the spurious element, Figure 8.18. Thus, selectively launching second generation waves at shock loop effectively removes spurious edge element and recovers the appropriate symmetries, Figure 8.19 without generating an additional symmetries, Figure 8.19e. Figure 8.20 shows a similar example of the removal of a spurious edge

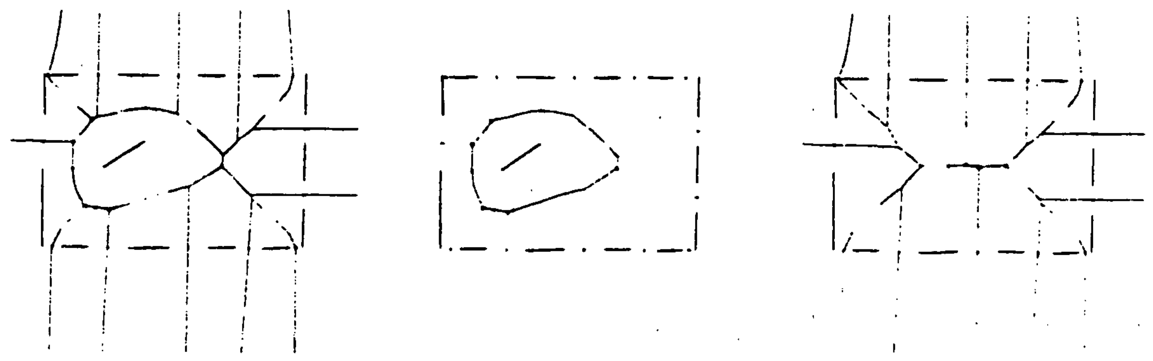


Figure 8.20: The existence of a loop in the shock structure signifies the potential existence of an isolated edge element. as shown in the synthetic edge map (d). The removal of this (potentially spurious) element effectively removes the loop and vice-versa. The removal of the loop highlighted in (e) requires the propagation of waves from the outside of the loop. Of course. the question of which transform to apply and in what order rests in the global selection of the least action sequence of transforms.

element. Contour grouping via gap transforms often requires a notion of inter-penetrating waves and saliency, Figure 8.21.

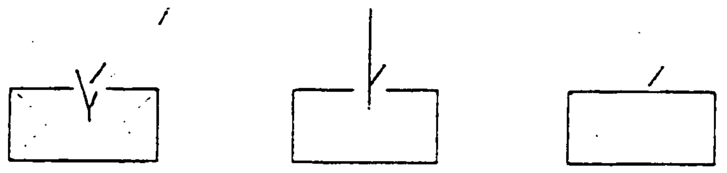


Figure 8.21: (a) a spurious edge element's interference with contour grouping can be removed by considering multiple generation of shocks as shown in (b) where second generation shocks arising from the shocks loops (completed by the image boundary) generate a new grouping hypothesis thus completing the rectangle symmetries (c).

8.4 Non-Blending Occlusion Transforms

In the real world, objects that are deforming or moving often occlude other objects and *often* their projections onto 2D images do not blend with the background nor with the objects

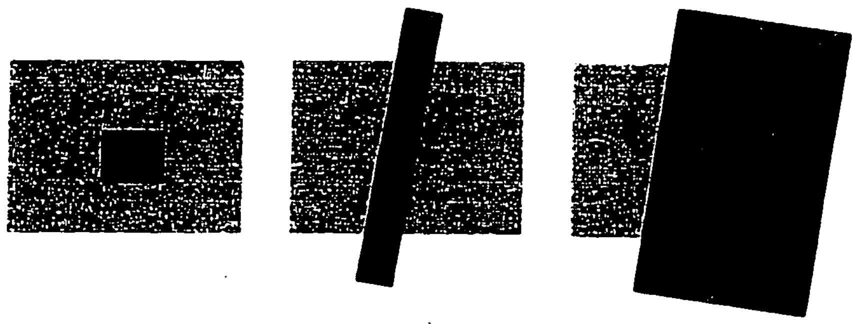


Figure 8.22: This figure depicts some possible cases of non-blending occlusions

that they occlude. Similarly, non-blending occlusions may stem from the photometric visual transformations, *e.g.*, change in the view, highlight, shadows, *etc.* In the case of these non-blending occlusions, the outline of the occluded objects may be distorted significantly, whereas the occluder experiences no significant changes in its boundary representation.

Let us now study the effects of a non-blending occlusion in the symmetry representation. Recall that occlusions are very similar to spurious elements because they corresponds to the addition of new boundary segments to edge map. In fact, if the occluder is totally inside the occluded object and if their boundaries do not intersect, the distorted symmetries can be corrected by a loop transform. Recall that loop transform can be seen as removing the spurious (or unwanted) element from the edge map. However, not every occlusion results in a loop structure in the symmetry representation. In many cases, an occluder may occlude more than one object, and a portion of background and may be partially occluded with other objects. In these cases, the the occluder boundary and the occluded object boundary intersect at some places, thus forming the junctions, Figure 8.22. The determination of an occluder, which is also known as the depth segregation task [145], can be done by detecting boundary junctions and then selecting boundary segments which connects smoothly at junctions as the boundary of the occluder. We define non-blending occlusion transform as a symmetry transform which removes the effects of the occluder on the symmetry axes.

Definition 9 The non-blending occlusion transform *corrects the distorted symmetries due to a non-blending occlusion by re-initializing waves from the shock (symmetry) branches, which are formed from the occluder boundaries.*

Figure 8.23 illustrates the non-blending occlusion transform acting on a synthetic shape. When the occluder is identified (separate graph) and is removed from the scene gaps are left in the object boundaries which are then subjected to a gap transform, Figure 8.23 (bottom).

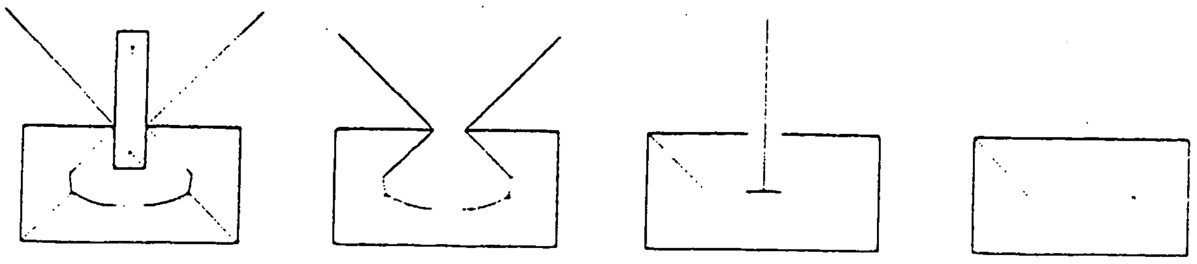
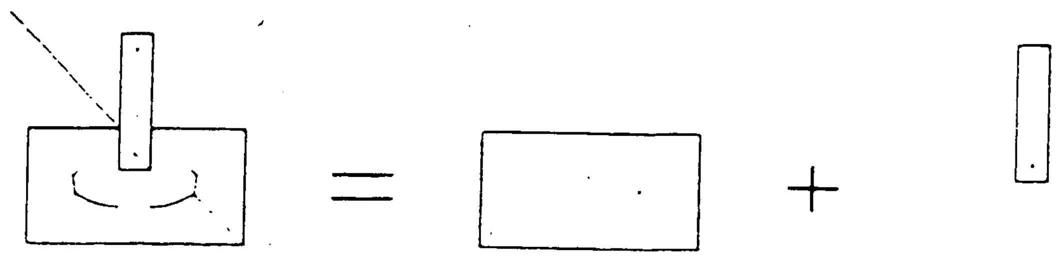



Figure 8.23: The figure depicts that the occlusion task can be solved by a first step of depth segregation, second a non-blending occlusion transform and finally a gap transform. (TOP) The non-blending occlusion transform is shown in detail. (left) The symmetry map of a scene containing an occlusion. (middle) The symmetry segments that are formed from the identified occluder are determined and shown in. (right) Secondary waves are initiated from the selected symmetries to correct the distorted symmetries due to the occluder. Note that the removal of occluder leaves a gap on the object boundary which may now be closed by the occlusion transform. (BOTTOM) The reconstruction of the scene by occlusion transform followed by a gap transform.

8.5 Canonical Operators For Symmetry Transforms

These symmetry transforms can be implemented as a combination of two canonical operators. First, observe that the gap/part transform requires the deletion of the effects of removing a boundary segment followed by the addition of the effect of a new one (for the gap transform, this represents a null part) on the symmetry set. Second, the loop transform requires the removal of the effects of the spurious element on the symmetry set, followed by asserting the effect of the existing boundary elements. Both transforms can be implemented in a brute force manner by recomputing the entire symmetry map after each deletion/addition of boundary segments. However, fortunately, the computation of symmetry points is local in nature, *i.e.*, the domain of dependence of shock point, or the range of influence of a boundary point, is typically a finite segment. Thus, in the deletion and

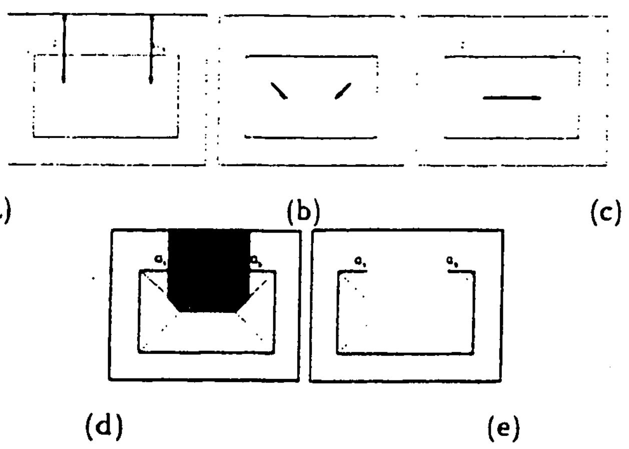


Figure 8.24: The deletion of a boundary segment Q_1Q_2 requires that its influence zone be identified and its effect be removed. Observe that only the characteristics from end points need to be propagated (a) as long as they are also propagated along shock paths. (b) and (c). thus simulating the effect of propagating characteristics from each point of the line segment. The resulting influence zone and the symmetry map after removing shocks in the influence zone (e) are shown. The final result also stresses that the deletion operator does not produce a final symmetry map: rather it must be combined with the addition operator to result in a meaningful symmetry map.

addition of boundary segments, appropriate regions can be identified. thus requiring only a minimal re-computation for obtaining the symmetry set after deletion or addition of a boundary segment. We must stress that the canonical operators are not in isolation of the appropriate symmetry transforms, but only constitute an element of a transform.

8.5.1 Deletion Operator

This operator removes the effect of a boundary segment on the symmetry map. Suppose that a boundary segment between two points, Q_1 and Q_2, is to be deleted from the boundary representation, Figure 8.24. Observe that in wave propagation, governed by by a hyperbolic PDE, a point on the initial boundary can only influence a region of the solution domain. *the range of influence (or influence zone).* The influence zone of a point on a regular curve can be determined by moving along the characteristics from that initial point up to where they terminate at shocks. Similarly, the influence zone of a finite boundary segment can be easily determined by propagating along the end point rays *and* the shockwaves that form from them, Figure 8.24, thus avoiding propagation of characteristics from every point of the boundary segment.

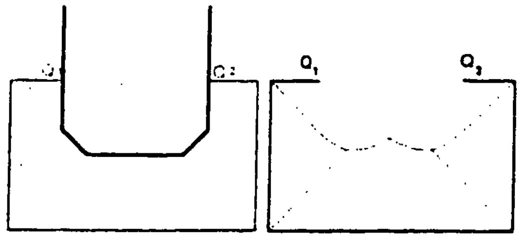


Figure 8.25: The addition operator fills without a contour (gap or null part) the "vacuum" left behind by the deletion operator in the symmetry map, Figure 8.24. In this figure, the effect of adding a null boundary is illustrated, which means propagating waves from points Q_1 and Q_2, in effect requiring propagation of waves from the boundary of influence zone.

8.5.2 Addition Operator

This operator addresses the effect of adding a curve segment to the existing edge map. Again, the influence zone of this boundary segment corresponds to the region where the changes on the symmetry set will occur. The key task is the correct and efficient determination of the influence zone. This is determined by initiating discrete waves from the added boundary segment, thus recreating the underlying symmetry map and associated waves. Specifically, this is done in two ways: (*i*) discrete waves overwrite waves with time of arrival greater than current wave, (*ii*) when the shockwaves originated from the new boundary segment quench with previous shockwaves, new shockwaves form and the affected shockwaves (and their children) are modified or deleted. Figure 8.25 shows the addition operator when a null part (gap) is added to the result of Figure 8.24.

We can describe each symmetry transform as a combination of these canonical operations. the gap/part transform can be expressed as a combination of deletion and addition operators: the completion of a gap, Figure 8.26, requires deleting the effect of the gap (null part) on the symmetry set and adding the effect of the gap completion boundary. Similarly, the part transform requires deleting a (correct) boundary segment and replacing it with another, Figure 8.10. Similarly, the loop transform removes the effect of an edge segment: observe that the influence zone of an edge element is precisely the loop itself. Thus, the loop transform is a combined deletion and addition operator: the deletion operator removes the symmetries inside of the loop, while the addition operator (in this case a null segment is added) fills the vacuum by propagating waves from the boundaries of the influence zone, *i.e.*, the loop, Figure 8.18.

In summary, we have proposed the symmetry map and associated transforms as a lan-

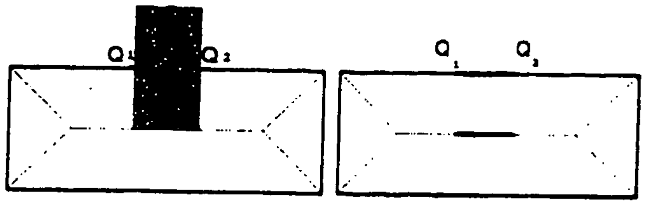


Figure 8.26: The application of an gap/part transform for gap completion: The transform consists of a deletion operator (left) to remove the effect of the gap on the symmetry map and (right) to add the affect of adding the gap completion boundary.

guage to explicitly represent perceptual grouping operations. While we have described the gap/part and loop transforms. we have not visited the implications of the splice transform. discovered as a transition of closed shape. Due to its significance as a boundary smoothing operator, we discuss this in a separate chapter.

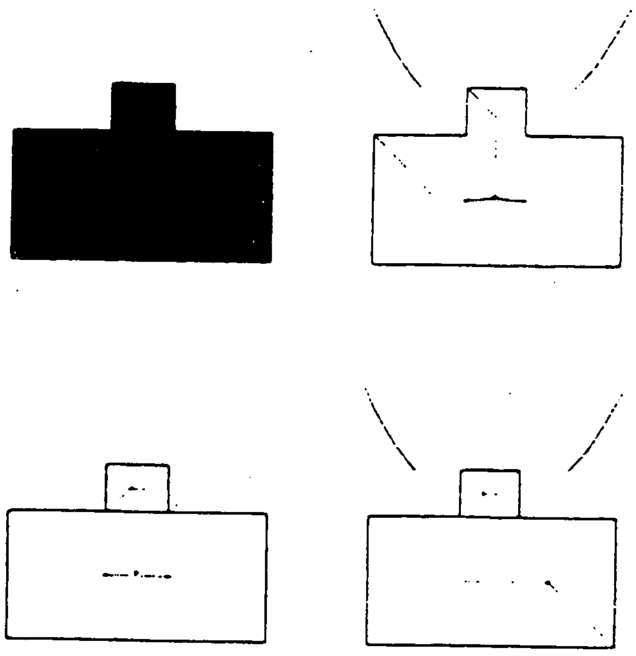


Figure 8.27: The application of an gap/part transform for part computation: The transform consists of a deletion operator to remove the effect of the corner points and a addition operator to add the affect of the adding the partline.

Chapter 9

Boundary Smoothing

The main idea in boundary smoothing is to remove noise present in the object boundaries without affecting the important features. In general, object boundaries are very noisy. The main sources of noise are visual transformations, non-linearities in image acquisition devices, and limitations of low-level vision algorithms. It is well accepted that boundary smoothing is an important step in image understanding because it is the basis for image processing operations, such as edge grouping shape decomposition, shape recovery, *etc.* For example, in edge grouping, noisy edge segments alter salience computations. Similarly, boundary partitioning schemes based on limbs [195] face with severe difficulties in the presence of noise since noisy object boundaries often contain many curvature minima and cause errors in geometric measurements, *e.g.* computation of tangent vector of noisy curves, are often unreliable.

Most current boundary smoothing methods are based on smoothing object boundaries via local operators, *e.g.* Gaussian, non-linear diffusion, morphological, *etc.* Also, the grey-level images are often smoothed by these local smoothing operators before a segmentation algorithm is applied. Unless the output of segmentation algorithms is a geometric boundary model, an additional boundary smoothing may be required to remove the discretization effects since many algorithms, *e.g,* shape decomposition, grouping edges, require smooth boundary representation. The main disadvantage of smoothing via local operators is the removal of important boundary features during the smoothing process since it is difficult to distinguish the important features from noise via local operators in a such an early stage of shape analysis.

Regularization of symmetry axes is an alternative approach to the boundary smoothing problem. The main idea stems from the fact that removal of an axis point corresponds to the smoothing of a boundary segment. Previously, most approaches *pruned* axis points whose saliency measures fall below a certain threshold. Any operations on the symmetry set should result in a valid symmetry map. The removal of single symmetry point has implications on other points of the symmetry map. Traditionally, these have not been considered and constitute a major shortcoming.

In this paper, we propose that pruning should operate on the symmetry *branches*, instead of axis point, since the removal of a branch corresponds to a structural change in the shape space (transition). Also, note that the removal of an axis branch modifies (or smooths) a portion of a boundary segment, thus any symmetry axis formed from this modified boundary segment (on the other side of the contour) is no longer valid. Thus, we present a coupled symmetry/boundary smoothing technique for curve smoothing. Specifically, our approach consists of three steps: (*i*) remove an axis branch with the smallest saliency measure, (*ii*) reconstruct the smoothed boundary segment due to the removal of the axis branch in (*i*), (*iii*) recompute the symmetry axes and their saliency measurements. In this approach, we use the area removed from the shape due to the pruning an axes branch as the saliency measure of the axis branches. This approach is also computationally efficient since the reconstruction of the boundary and re-computation of the symmetries are done locally by canonical shock operators presented in the previous chapter. We show that our approach preserves important boundary features, *e.g.* corners, and groups boundary segments together.

In Section 9.1, we first review smoothing techniques based on local filters. Specifically, we summarize (*i*) Gaussian operator based approaches, (*ii*) non-linear geometric diffusion methods and (*iii*) morphological transformations, as well as the application of the discrete wave propagation method proposed in Chapter 4 for smoothing open curve segments as well as shapes. Section 9.2 presents a novel work on the use of the splice transform for boundary smoothing.

9.1 Smoothing Via Local Operators

Most current shape (or image) smoothing algorithms are based on filtering the shape (or image) by some *local operators*. These local operators can be classified into three main groups: (*i*) the Gaussian operators (linear operator), (*ii*) nonlinear diffusion operators, (*iii*) morphological operators. We now summarize these approaches in detail.

9.1.1 Gaussian Filtering

The Gaussian filters of different sizes have been extensively used in shape and image smoothing. Marr and Hildreth [136] used the Gaussian operators to find the edges of gray-scale images. Specifically, the edges are detected as the zero-crossings of Laplacian of images convolved with Gaussian filters of different sizes. Witkin [235] proposed a *scale-space* concept in which 1D boundary signals are convolved with different sizes of Gaussian and the zeros of the second derivative are localized and tracked as the size of filter increases. It was noted that no new zero-crossings are created as the scale increases. Similar results for 2D images were presented in [240, 17]. Recently, Lindeberg [127] proposed that significant image structures can be detected from the Gaussian scale-space representation. Koenderink [113, 115] showed that the Gaussian scale-space can be modeled by the heat diffusion partial differential equation,

$$u_t = u_{xx} + u_{yy} \tag{9.1}$$

where u represents the image, x, y the spatial dimensions, and t specifies the scale parameter (the amount of smoothing).

Asada and Brady [13] presented five boundary primitives based on the curvature changes of shape boundaries smoothed by Gaussian. In this *primal curvature sketch* method, the curvature changes are detected at each scale along the boundary and are then used as a set of knot points in a spline approximation to the contour. Similarly, zero-crossings of the curvature of a boundary curve is used by Mokhtarian and Mackworth [138, 139]. Specifically, they proposed a curvature-scale space in which shapes are increasingly smoothed by Gaussian and then their inflection points are tracked for matching.

There are two major shortcomings of Gaussian smoothing: (i) Gaussian smoothing shrinks shapes and dislocates boundaries when moving from finer to coarser scales. (ii) Gaussian smoothing blurs important image features. Weickert [231] noted that any approach to overcome these difficulties is at the expanse of renouncing linearity or some other scale-space properties. The shrinkage problem of Gaussian filtering was addressed in [85, 131, 130, 151, 117]. Specifically, Horn and Weldon [85] proposed to filter the extended circular image of the curve with a Gaussian filter. In [131, 151] the shrinkage problem was compensated by inflating the curve proportional to the curvature of smoothed

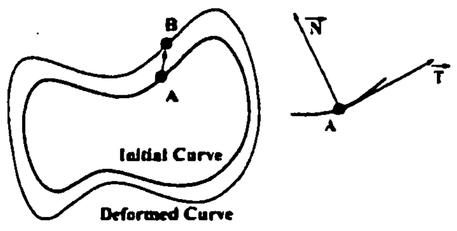


Figure 9.1: The points on the initial curve A move to B to generate a new curve. The direction and magnitude of this motion is arbitrary in order to capture general deformations [106].

curve. Latecki and Lakamper [117] present a discrete contour evolution of polygonal shapes in which vertices are recursively deleted to preserve the location of important boundary features. Perona and Malik presented a nonlinear diffusion filtering method which reduces smoothing at the edges to preserve the contrast information and the location of the object boundaries. Specifically, they proposed

$$\begin{cases} u_t = div(f(|\nabla u|^2)\nabla u) \\ u(x,y,0) = I(x,y) \end{cases} \tag{9.2}$$

where $I(x,y)$ is the image and $f(|\nabla u|^2) = \frac{1}{1+|\nabla u|^2/\lambda^2}$ and $\lambda \geq 0$. This method preserves the strong edges for a long time during smoothing process by turning off the smoothing at high brightness gradient locations. Thus, this approach smooths regions of low brightness gradient while regions of high gradients are not smoothed. Catte *et. al.* [38] suggest a slight modification in which the gradient in $f(|\nabla u|^2)$ is computed from the Gaussian smoothed image to account the large noise in the image

$$u_t = div(f(|\nabla(u * G_\sigma)|)\nabla u) \tag{9.3}$$

Similar scheme is proposed by Whitaker and Pizer [233]. However, these anisotropic diffusion approaches are contrast-driven and smooth globally salient but low contrast image features [104]. Weickert presented an excellent review of the related approaches in [231].

9.1.2 Nonlinear Geometry Driven Diffusion

Kimia [99] has proposed a shape framework based on the observation that slightly deformed shapes are visually similar. Specifically, Kimia *et al.* [105, 107, 106, 108, 112] studied shape in the context of deformations which depend on the local geometry of shapes. The local deformations are formulated as the sum of a constant motion and motion proportional to the curvature at that point. Consider a curve $C_0(s,t) = (x_0(s), y_0(s))$ undergoing a deformation, where s is the parameter along the curve, x_0 and y_0 are the Cartesian coordinates and the initial curve is denoted by the subscript o. Suppose also that each point on this curve move by some arbitrary amount in the outward normal $\vec{N}$, Figure 9.1 as follows

$$\begin{cases} \frac{\partial C}{\partial t} = \beta(\kappa(s))\vec{N} \\ C(s,0) = C_0(s) \end{cases} \tag{9.4}$$

where $\beta(\kappa(s))$ is an arbitrary function dependent on the local geometry of the curve at that point and κ is the curvature. Note that $\beta(\kappa(s))$ is independent of time (t) since it is reasonable to require that deformations do not depend on the time of the deformations. Kimia *et al.* chose a simple "first-order" model which captures both morphological operations as well as smoothing processes, - that is

$$\begin{cases} \frac{\partial C}{\partial t} = (\beta_0 - \beta_1\kappa)\vec{N} \\ C(s,0) = C_0(s) \end{cases} \tag{9.5}$$

The above PDE contains two types of deformations: constant deformation and curvature deformation. A pure constant deformation called *reaction* process, $\frac{\partial C}{\partial t} = \beta_0\vec{N}$ may create singularities (corners) even when the initial curve is smooth, Chapter 1. The second term in (9.5) is called diffusion term. The pure diffusion process

$$\frac{\partial C}{\partial t} = -\kappa\vec{N} \tag{9.6}$$

has been called *geometric heat equation*. This equation is a parabolic equation, thus it does not create any singularities during the deformation process. Gage and Hamilton [68, 70, 69] showed that a convex curve undergoing a curvature evolution goes to a round point without self-intersections. Grayson [77] generalized this result to show that *any* embedded curve will become convex without developing self-intersections. It should be noted that the geometric heat equation is invariant to the Euclidean transformations (rotation, translation, and scaling). Sapiro and Tannenbaum [174] extended this invariance to affine transformations by introducing an affine invariant geometric heat flow

$$\frac{\partial C}{\partial t} = -\kappa^{1/3}\vec{N} \tag{9.7}$$

The affine flow shrinks any smooth non-convex curve to an elliptical point.

Kimia *et al.* [107, 106, 112] have presented a reaction-diffusion space for shapes as the solution of (9.5) for different values of β_0/β_1. It was noted that the numerical implementation of this reaction-diffusion equation is a difficult task due to the (i) singularities that

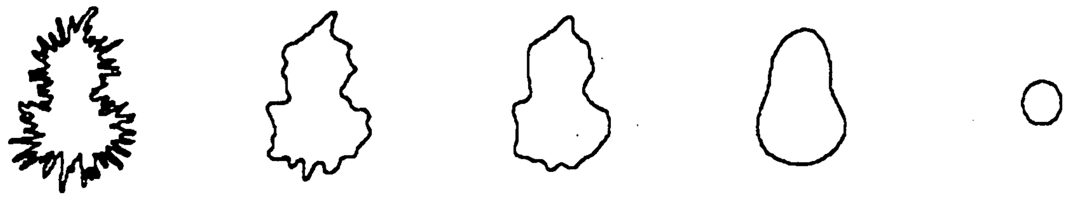


Figure 9.2: This figure depicts the diffusion process on a noisy shape. Note that the noisy shape boundaries shrinks to a circular shape.

form in reaction process, (*ii*) topological changes during the deformation. (*iii*) numerical errors in geometric measurements, *e.g.*, curvature, gradient. The remedy for this problem is to embed the evolving curve, $C(s,t)$ as the level set of an evolving surface [154, 20]. Specifically, the evolving curve is represented by the zero level set of a hypersurface $\phi(x,y,t) = 0$. The zero level set of surfaces evolving according to

$$\phi_t = \beta_0|\nabla\phi| - \beta_1 div(\frac{\nabla\phi}{|\nabla\phi|})|\nabla\phi| \tag{9.8}$$

corresponds to the viscosity solutions of (9.5) as shown in [154, 184, 182, 183, 112]. Kimia *et al.* have noted that reaction process of shapes provides information about the parts and protrusions of shapes while diffusion process smooths the boundary of the shape.

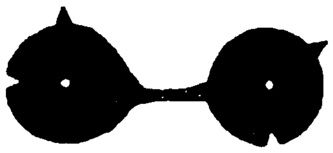
The curvature smoothing defined for the shapes has easily been extended to the grey-level images since any continuous surface, $\phi(x,y)$ can be used as an evolving surface in (9.8). Thus, grey-level intensity or range information can be used as a evolving surface, $\phi(x,y,t)$. The curvature deformation, which is also known as the mean curvature flow.

$$\phi_t = div(\frac{\nabla\phi}{|\nabla\phi|})|\nabla\phi| \tag{9.9}$$

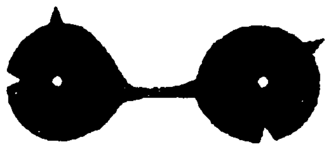
was studied by Evans and Spruck [61, 62, 63] and Chen *et al.*[41]. Kimia and Siddiqi [104] also used this evolution for image smoothing. Similarly, Alvarez *et al.*[6] studied a class of nonlinear parabolic differential equations defined by

$$\begin{cases} \phi_t = g(|G * \nabla\phi)div(\frac{\nabla\phi}{|\nabla\phi|})|\nabla\phi| \\ \phi(x,y,0) = \phi_0(x,y) \end{cases} \tag{9.10}$$

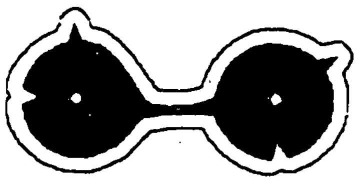
where G is a smoothing kernel, *e.g.* Gaussian. This is similar to Perona Malik's anisotropic smoothing in that curvature smoothing is turned off in the vicinity of a high brightness gradient. Osher and Rudin [153] proposed the shock filters for image smoothing and enhancement. Recently, Malladi and Sethian considered flows under min/max curvature [132]. The other works on the nonlinear geometric diffusion can be found in [220, 221].



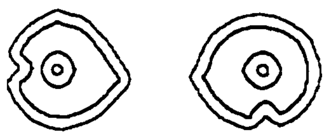
(a) original shape



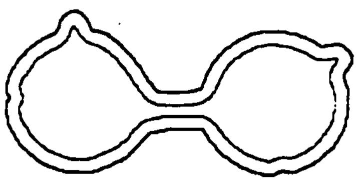
(b) erosion



(b) dilation



(d) opening



(e) closing

Figure 9.3: The morphological operators are applied on a test shape (a). The shape boundary after erosion (b) and dilations (c) transformations are shown in red. Similarly, the shape boundary after (d) opening and (e) closing operators are used to smooth the shape. Note that the opening operator removes small protrusions and holes and breaks shapes into parts in narrow regions, whereas the closing operator removes small indentations.

9.1.3 Morphological Boundary Smoothing

Classically, mathematical morphology treats binary images as sets and grey-scale images as functions and operates on them in the spatial domain via *morphological transformations* using *structuring elements* [137, 179, 79, 180]. For a given binary image represented by the set X and a structuring element B, two elementary morphological operators are the dilation, $\oplus$ and the erosion, $\ominus$:

Definition 10 *The dilation of a set X by the structuring element B is defined as*

$$\begin{aligned} X \oplus B &= \{x - b \mid x \in X, b \in B\} \\ &= \bigcup_{b \in B} X_{-b}. \end{aligned}$$

where X_{-b} is the translation of the set X by $-b$.

Definition 11 *The erosion of a set X by the structuring element B is defined as*

$$\begin{aligned} X \ominus B &= \{x \in \mathbf{R}^2 \mid B_x \subseteq X\} \\ &= \bigcap_{b \in B} X_{+b}. \end{aligned}$$

where X_{+b} is the translation of the set X by $+b$.

Note that the erosion of a set X with B can be computed by dilating the the complement X^C by B and take the complement of the result. Figure 9.3 illustrates the dilation and erosion of a shape by a disk structuring element. Combining erosions and dilations yields two other important morphological filters: the opening and the closing which are given by

$$\begin{aligned} \gamma_B(X) &= (X \ominus B) \oplus B. \\ \phi_B(X) &= (X \oplus B) \ominus B. \end{aligned}$$

As illustrated in Figure 9.3, for shape smoothing opening tend to remove sharp protrusions and breaks shape into parts in narrow regions, whereas closing tend to fill in small holes and gaps.

Historically, morphology transformations are applied to objects with discrete boundary representation. Recently, there has been an interest in considering *geometric interpretations* of morphological transformations, where basic morphological operators can be formulated as partial differential equations governing the geometric evolution of shapes. To illustrate, consider the dilation transformation of a shape with a disk structuring element. The transformed shape is the union of all disks centered on points of the original shape. The boundary of the transformed shape is a curve parallel to the boundary of the original shape with a distance equal to the radius of the disk. This is precisely Huygens' principle for wavefront propagation [94], relating operations on algebraic constructs, *i.e.*, sets of points, on the one hand, and operations on geometric entities, *i.e.*, curves representing the boundary on the other. Can this deformation of a curve onto another as a result of the dilation be described as a sequence of infinitesimal deformations each of which is only locally dependent on the original boundary's shape, and thus represented by partial differential equations?

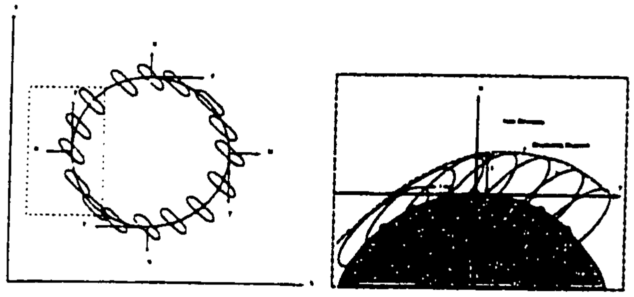


Figure 9.4: Morphological operations in the extrinsic and intrinsic coordinates for an arbitrary convex structuring element.

9.1.4 Mathematical Morphology and Curve Evolution

The answer in certain circumstances is positive and founded on three observations. First, observe that for a class of structuring elements, $\hat{\mathcal{B}}$, n repeated dilations (or erosions) with a structuring element $\mathcal{B} \in \hat{\mathcal{B}}$, is equivalent to a single dilation with a structuring element of the same shape, but which is "scaled up" n times, $\mathcal{B}(\lambda)$, *e.g.*, 10 dilations with a circle of radius of 1, $\mathcal{B}(1)$, is exactly a single dilation with a circle of radius 10, $\mathcal{B}(10)$. The class of structuring elements $\hat{\mathcal{B}}$ for which this property holds is exactly the set of *convex* structuring elements [137, Theorem 1.5.1, pp. 22–23]. Now, conversely, given a single operation with a convex structuring element, the operation can be decomposed as a sequence of n operations with the same structuring element but of "size" $1/n^{th}$ the original. In the limit, as $n \to \infty$, this notion of size constitutes the "time" axis for the differential deformation. Second, observe that the outcome of mathematical morphology operations with a shape $\mathcal{S}$, can be determined solely from its boundary, $\partial\mathcal{S}$ [228]. Thus, the mapping from the shape $\mathcal{S}$ to $\mathcal{S}'$ by morphological operations can be reduced to operations taking $\partial\mathcal{S}$ to $\partial\mathcal{S}'$. Third, for a sufficiently small but finite structuring elements, mathematical morphology operations along the smooth portions of the boundary can be viewed as a deformation along the normal by a certain amount, Figure 9.1. Formally, let some shape $\mathcal{C}(s,t)$ evolve by a mathematical morphology operation, say dilation, by some structuring element $\mathcal{B}(\lambda)$ of size λ, to a new shape $\mathcal{C}(s,t+\lambda)$. This evolution can be represented as $\mathcal{C}(s,t+\lambda) - \mathcal{C}(s,t) = \Gamma(s,t,\lambda)\vec{N}(s,t)$, where Γ is the distance along $\vec{N}$ that a point on the boundary moves in a dilation operation with a structuring element of size λ. Since $\Gamma(s,t,0) = 0$,

$$\frac{\partial \mathcal{C}}{\partial t} = \lim_{\lambda \to 0} \frac{\Gamma(s,t,\lambda) - \Gamma(s,t,0)}{\lambda}\vec{N} = \frac{\partial \Gamma(s,t,\lambda)}{\partial \lambda}|_{\lambda=0}\vec{N} = \beta(s,t)\vec{N}. \qquad (9.11)$$

It has been shown that the amount of differential deformation, β, of a shape $\mathcal{S}$ at a point P due to dilation (erosion) with a convex structuring element $\mathcal{B}$, is the maximal (minimal) projection of $\mathcal{B}$ onto the normal $\vec{N}$ of the boundary at P [11, 12], so that β can only depend on the orientation of the target to the shape $\vec{T}(s,t)$, leading to

$$\begin{cases} \frac{\partial C}{\partial t} & = \beta(\vec{T}(s,t))\vec{N} \\ C(s,0) & = C_0(s), \end{cases} \tag{9.12}$$

or $\frac{\partial C}{\partial t} = \beta(\theta)\vec{N}$ where $\theta = \angle(\vec{T}, x\text{-axis})$. The erosion transformation is similarly modeled.

Related Work: Similar differential equations were derived by Brockett and Maragos [30] for the evolution of signals $f(x)$ which dilate (or erode) by functions $g(x)$ or structuring element B whose domain and range is scaled by t. Specifically, for 1D sets $\frac{\partial a(x,t)}{\partial t} = \pm|\frac{\partial a(x,t)}{\partial x}|$ and for 2D sets $\frac{\partial a(x,y,t)}{\partial t}$ is equal to $(|\alpha_x|^2+|\alpha_y|^2)$ for a disk. $= (|\alpha_x|+|\alpha_y|)$ for a square. and $= max(|\alpha_x|, |\alpha_y|)$ for a diamond. They also described evolution equations for dilations/erosions by functions.

Alvarez, Guichard, Lions, and Morel [5] proved that under certain initial conditions required for a multi-scale analysis T_t transforming an intuitive $u_o(x)$ into $u(x,t) = (T_t u_0)(x)$. the smoothed signal must satisfy a partial differential equation

$$\frac{\partial u}{\partial t} = F(D^2 u, \nabla u, t) \tag{9.13}$$

where D^2 is a second-order differential operator and ∇ is the gradient. For example.

$$\frac{\partial u}{\partial t} = \pm|\nabla u| \tag{9.14}$$

represents dilation and erosion. As another example, Catte, Dibos and Koepfler [37] cast Euclidean curve shortening flow $\frac{\partial C}{\partial t} = \kappa\vec{N}$. also known as mean curvature motion. in the mathematical morphology framework by considering a line segment structuring element of length $2\sqrt{2t}$, orientation $\theta\epsilon[0,\pi]$ and three operators

$$\begin{cases} I_t u(x) = \inf_{\theta\in[0,\pi]} \sup_{y\in X+S(\theta,t)} u(x), \\ S_t u(x) = \sup_{\theta\in[0,\pi]} \inf_{y\in X+S(\theta,t)} u(x), \\ C_t u(x) = \frac{1}{2}[(S_{2t}u)(x) + (I_{2t}u)(x)] \end{cases} \tag{9.15}$$

and show that these operators correspond to

$$\begin{cases} \frac{\partial C}{\partial t} = -\kappa^-\vec{N}, \\ \frac{\partial C}{\partial t} = \kappa^+\vec{N}, \\ \frac{\partial C}{\partial t} = \kappa\vec{N} \end{cases} \tag{9.16}$$

respectively. Boomgard's work is another example relating morphology and PDE approaches [56].

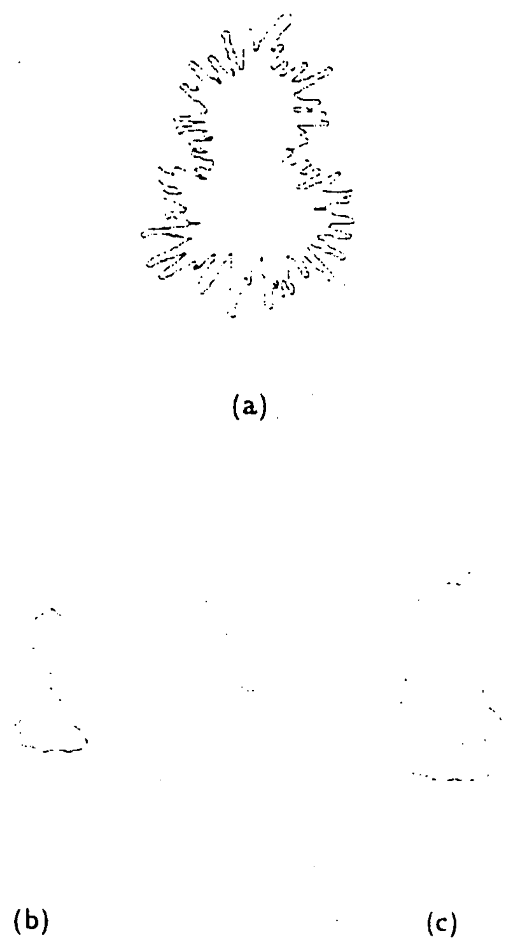


Figure 9.5: An example of the opening operation using discrete wave propagation algorithm. (a) the original shape. (b) the erosion operation. (c) opening operation.

9.1.5 Discrete Wave Propagation For Morphological Transformations

While the implementation of curve evolution as zero level sets of an embedding evolving surface is robust and even has subpixel ability [198], it cannot handle *junctions* and *open curves*, and is also *computationally inefficient* due to the additional embedding dimension. In Chapter 4, we presented *discrete wave propagation* approach [215] relying on a view of curve evolution as propagation of waves from boundary segments, inspired by Blum's "grass fire" [24]. In this approach, each boundary initiates waves carrying geometrical information, labels, *etc.* which if traveling with constant speed yields the *distance transform*, among other constructs. Recall that the distance transform maps a binary image onto an image where the value at each point is the Euclidean distance from the object, and thus distance represents "time" and a constant distance set represents a wavefront. We showed that this wave

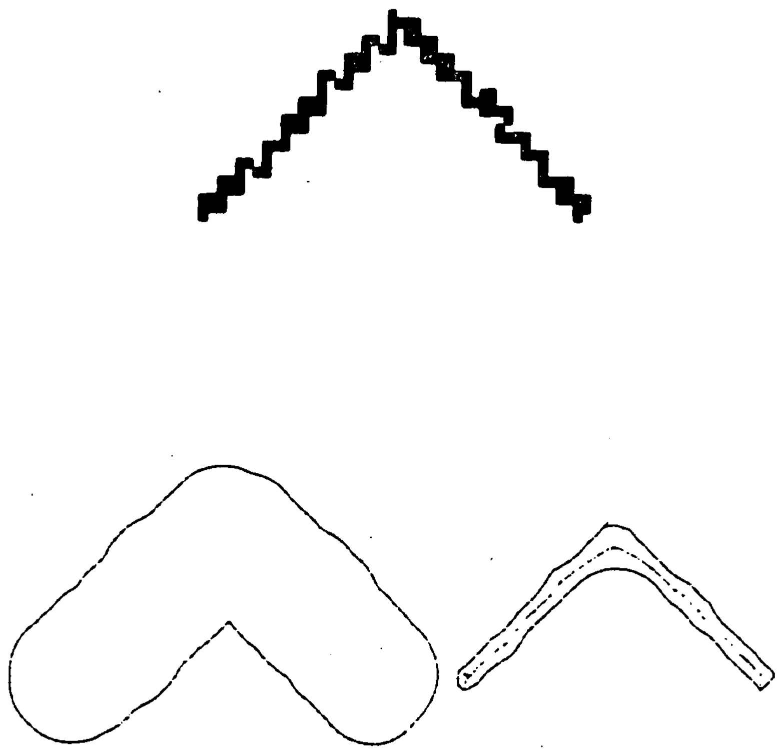


Figure 9.6: Smoothing open curves by subpixel closing. Observe how the location and orientation of the line is maintained to subpixel accuracy.

propagation approach based on distance transform can be robustly and reliably applied to subpixel curves. but implemented on a discrete grid with discrete directions. Unlike curve evolution, this approach does not require that shapes be segmented from images but rather is easily applied to open contour segments, possibly with junctions. as available in edge maps. In addition, this approach is computationally efficient since it does not require the higher-dimensional embedding surface.

Figure 9.5 illustrates smoothing of a shape by an opening operator implemented by discrete wave propagation. Similarly, a noisy curve segment is smoothed by a closing operator. Interestingly, the symmetry map of the smoothed boundary (depicted in red) may provide a better smoothing approach for the curve segments.

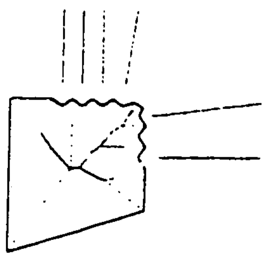


Figure 9.7: The small protrusions on the shape boundaries often result in long symmetry branches. These branches unfortunately also intersect with globally salient axes branches and break them into fragments.

9.2 Structural Smoothing

Symmetry axis representation is very sensitive to small changes in the shape boundary. Small perturbations of the boundary can lead to (*i*) the abrupt introduction of long branches, and (*ii*) the partitioning of the existing globally salient branches due to the introduction of these spurious branches, Figure 9.7. Unfortunately, the resulting symmetry map cannot be used in higher-level vision, *e.g.*, for object recognition or in symmetry transforms as defined in the previous chapter. Thus, the regularization of symmetry axes is a necessary step in order to use the elegant properties of this representation. Giblin and Kimia [75] classified the medial axis and shock instabilities for the closed curves. In this section, we consider the instabilities of the symmetry axes caused by small perturbations in the object boundary, and the corresponding splice transform.

Boundary smoothing presents a possible solution to the regularization of symmetry map. Previously, Gaussian smoothing of shapes was used by Dill *et al.* [57], and Pizer *et al.* [160] to construct a hierarchical description of symmetry axes. Similarly, Siddiqi and Kimia [196], and Tari and Shah [209] incorporated the curvature based-smoothing to their symmetry detection algorithms for the regularization of symmetry axes. However, boundary smoothing which is based on solely local geometry of boundary presents a remedy, but does not solve the problem. It is infact difficult to distinguish noise from the small details of boundary at such an early stage of shape representation [83, 149].

Alternative regularization approach is based on considering significance of each symmetry axis segment and organizing symmetries based on these saliency measurements. A popular approach which removes the symmetry axis segments based on a saliency measure associated with them is called *pruning.* The essential goal of pruning is to distinguish the symmetries that arise due to the globally salient features from that arise due to noise (or artifacts). Most previous approaches try to achieve this goal by requiring that the pruning method must preserve the initial connectivity (or topology) of the symmetry axes *i.e.*, no gaps must be created in the symmetry axes. In general, most symmetry regularization methods involve two steps: (*i*) determination of a saliency measure for each symmetry axis points; (*ii*) a process which makes a decision how a saliency measurements can be used in reaching the proposed goal.

An appropriate saliency measure for each symmetry axis segment (or point) can be computed from: (*i*) the properties of the symmetry axis, such as radius, speed; (*ii*) the domain of dependence of corresponding symmetry axis segment, *e.g.*, boundary or region information. To preserve the connectivity in the pruning stage, it is required that the saliency function of a simple symmetry map with a single root have a single maxima which corresponds to the root of the symmetry tree.

The second step of the regularization process is to find a method that removes certain portion of the symmetry axes based on the saliency measurements and modifies the remaining symmetry set representation. The most popular pruning process is removing the symmetry points whose saliency values are less than a preset threshold [149, 190]. However, this regularization method removes axis points belonging to significant branches since it does not take advantage of the global information stored in the symmetry axes. Ho and Dyer [83] proposed that the pruning should operate on the branches of symmetry set representation. However, their approach also smooths important features while trying to remove insignificant axis branches because most globally salient branches are broken into fragments due to small perturbations on shape boundaries.

In this section, we propose a novel approach for regularizing the instabilities caused by small perturbations in the object boundaries. Our approach is based on the *splice transform* which removes the least salient symmetry branch at each iteration. Thus, this approach is in agreement with the shape space proposed by Kimia in which symmetry transforms punctuate global deformation paths and segment them into portions where the representation changes continuously with shape deformation. Specifically, at each stage of our regularization process: (*i*) the least salient symmetry branch is removed, (*ii*) a boundary segment determined by the domain of dependence of the pruned symmetry branch is smoothed, (*iii*) the symmetries in the influence zone o of the smoothed boundary segment

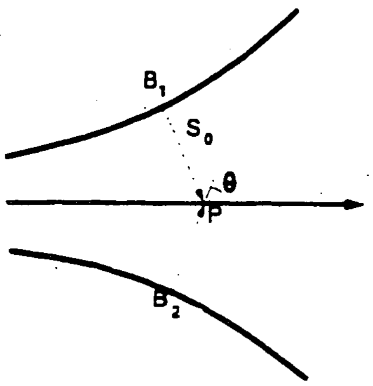


Figure 9.8: The speed of shock at point, P formed by the rays originated from boundary points B_1 and B_2 respectively can be computed from the speed of rays, s_0 and the quench angle, θ.

are re-computed. Our approach preserves important boundary features, like corners, groups unorganized curve segments and computationally efficient due to canonical operators.

9.2.1 Saliency of Symmetry Axes

The local characteristics of the symmetry axes, *e.g.* speed, radius have been previously used in the determination of a saliency function for symmetry maps [24, 140, 58, 9, 116, 123]. Specifically, the speed function for an axis point is given in [24, 140, 194]

$$v(\theta) = \frac{s_0}{cos(\theta)} \tag{9.17}$$

where s_0 is speed of the rays that form the shock (symmetry point), 2θ is the angle between these two rays, Figure 9.8. However, the speed function alone is not an appropriate saliency measure as the speed function does not necessarily have a single maxima for symmetries with a single root. For example, the speed function of a degenerate shock decreases as it propagates away from its origin, Figure 9.9. Thus, a pruning technique based solely on thresholding the speed function can create gaps in the symmetry map. In addition, the speed function is not an intuitive choice for saliency. Similarly, the radius function, which is again not intuitive, has been used for saliency measurements in [116].

Alternatively, the domain of dependence of a symmetry segment can be used to compute the saliency of a symmetry axis segment. Let us first summarize the saliency measures based on the boundary information. Figure 9.10 illustrates a simple boundary segment and its symmetry axis. Suppose that the saliency measure at an axis point, P, which is created

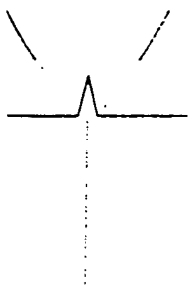


Figure 9.9: The speed of the degenerate shock branch decreases as it becomes longer since the angle between the rays coming from the corner points becomes smaller.

from the rays originating from the boundary points. A, B respectively is to be computed. The following definitions illustrated in Figure 9.10 are used in saliency measurements:

$$\begin{cases} bl_{AB} = & \text{lenght of boundary segment between } A \text{ and } B \\ ch_{AB} = & \text{the lenght of the chord between } A \text{ and } B \\ ca_{AB} = & \text{the length of the circular arc between } A \text{ and } B \text{ with radius } R \end{cases} \tag{9.18}$$

Blum and Nagel [25] observed that an introduction of a spurious symmetry branch due to a small protrusion in the boundary should not be interpreted as a significant branch in the symmetry set since it is created by a very small boundary segment. Their saliency measure for a shock branch is based on the length of the boundary segment unfolded by the branch and the length of the branch itself. Specifically, they proposed the boundary/axis ratio, $\frac{bl_{AB}}{length(P)}$ as a salience function, Figure 9.10, where $length(P)$ corresponds the total length of the axes branch from the origin to the axis point P.

Similarly, Ho and Dyer [83] proposed a saliency function based on the prominence of the boundary segment relative to the local radius and local smoothness of shape. Specifically, the saliency function is given by $\frac{d_2}{R}$ where d_2 (or d_1 in the case of a chord approximation) is the maximum distance between the boundary segment and the circular arc segment connecting the end points of the boundary segment and R is the radius of the point P in the symmetry axis, Figure 9.10. The distance, d_2, erosion thickness is also used in [28, 144, 173, 190]. Similarly, Ogniewicz and Kübler [149] have also proposed three different saliency functions based on the variations of the boundary length: (i) the boundary length itself, $bl(AB)$, (ii) the difference between between length of a boundary segment and its circular arc representation, $bl_{AB} - \frac{2}{\pi}ca_{AB}$, and ($ii$) the difference between between length of a boundary segment and its chord representation, $bl_{AB} - ch_{AB}$, Figure 9.10.

In this chapter, we propose that the domain of dependence of a symmetry axis segment

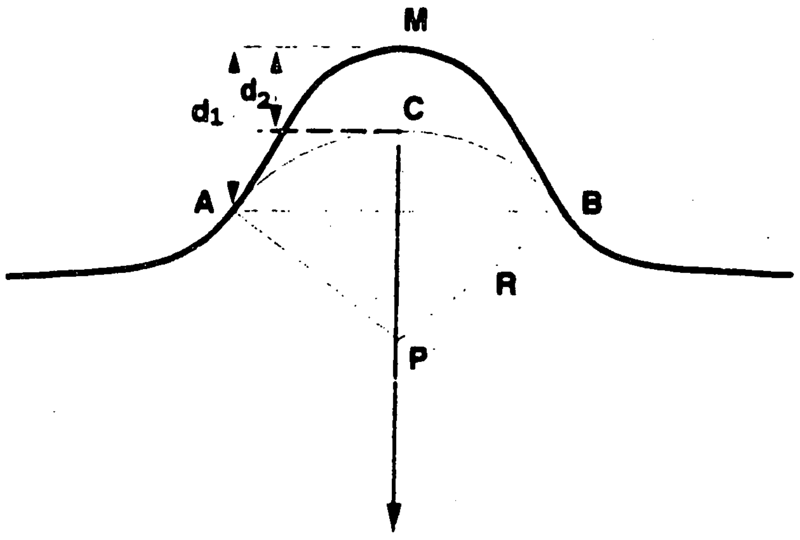


Figure 9.10: This figure illustrates that a saliency of a symmetry axis point, P can be measured by a difference function between original boundary, AB and its circular arc (or chord) approximations. For example, d_1 (or d_2) are proposed by Ho and Dyer [83]. Similarly, the length of boundary segment, w_{AB} (unfolded by the axis point P) and its approximations, *e.g.*, chord, circular arc are used in others.

provides crucial information about its saliency. Recall that for hyperbolic PDE's the domain of dependence (DOP) (or dependence zone) of a point (x, t) is defined as the boundary points that affect the solution at point (x, t). As summarized above, Blum and Nagel and others proposed to use the boundary segment contained in the DOP to measure the saliency. We advocate that the region in the domain of dependence of a symmetry axis is a better choice for saliency measurements than the boundary by itself as symmetry representation combines both boundary and regional information. This is also supported by the fact that the removal of a symmetry axis segment translates to the removal of a region from the object (or part of object). Previously, saliency measure based on the region information was first proposed by Cordella and di Baja [50] and used in [14, 190]. Figure 9.12 compares the area- and the boundary length-based saliency functions in a pruning process.

Let us illustrate this on the simple symmetry axis illustrated in Figure 9.11. For a given symmetry axis with the associated radius function, it is possible to reconstruct the boundary by an envelope of circles whose radii are specified in symmetry representation. Figure 9.11c illustrates that the reconstruction (starting from an axis point with the highest radius value) may be viewed as an iterative process in which a layer is added in each iteration. Alternatively, the reconstructed shape from its symmetry map gives clues about

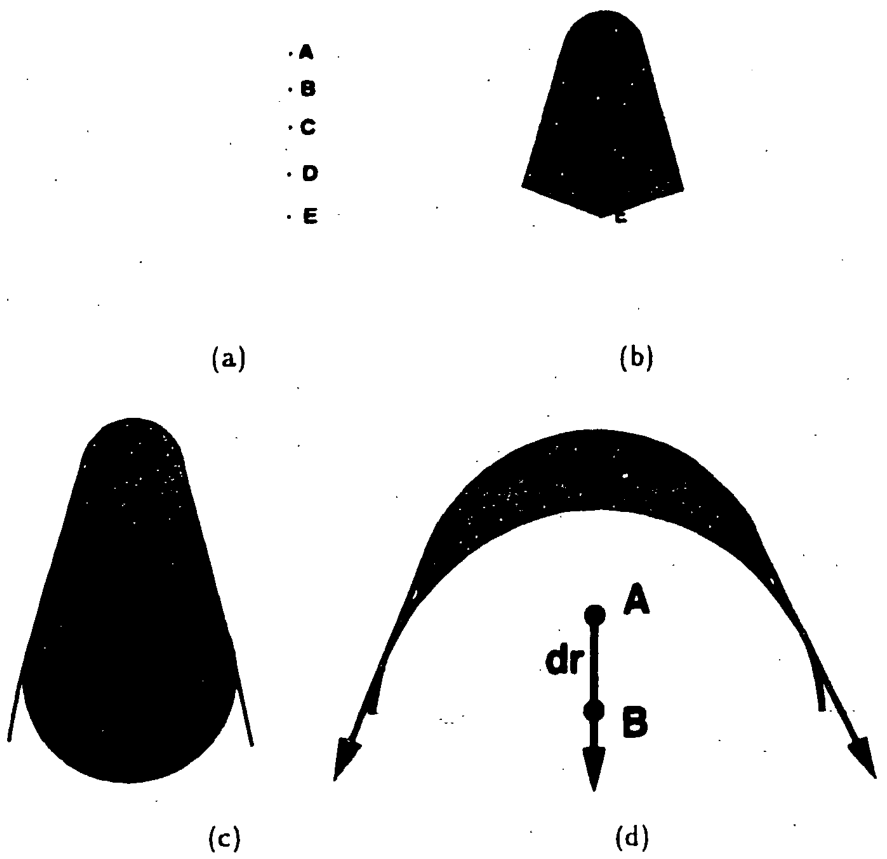


Figure 9.11: (a) A boundary segment and its partial symmetry axes contain five sample points. (b) The dependence zone of these axis branch between A and E is shown in gray. (c) The boundary of a symmetry axis can be reconstructed by the envelopes of circles whose radii are specified in symmetry representation. (d) a crescent region ΔA can be used to approximate the region that is peeled off from the shape due to the removal of a symmetry segment, Δr.

how a pruning should modify the shape representation. Specifically, removal of a small axis segment can be interpreted as peeling a layer off from the given shape. Thus, we propose that the saliency of an axis point can be computed from the region that is removed due to the removal of the axis segments. Figure 9.11d illustrates that a crescent region ΔA can be used to approximate the region that is peeled off from the shape due to the removal of a symmetry segment, Δr. This crescent differential area is given by [100, 190]

$$\Delta A = 2r(\sqrt{v^2 - 1}) + cos^{-1}(\frac{1}{v}))\Delta r. \tag{9.19}$$

where v is the speed of an axis point at B and r is the radius of B. Thus, we are now able to compute the area associated with a symmetry axis point by integrating these differential regions. Specifically, let us assume that a symmetry axis branch is represented by n piecewise

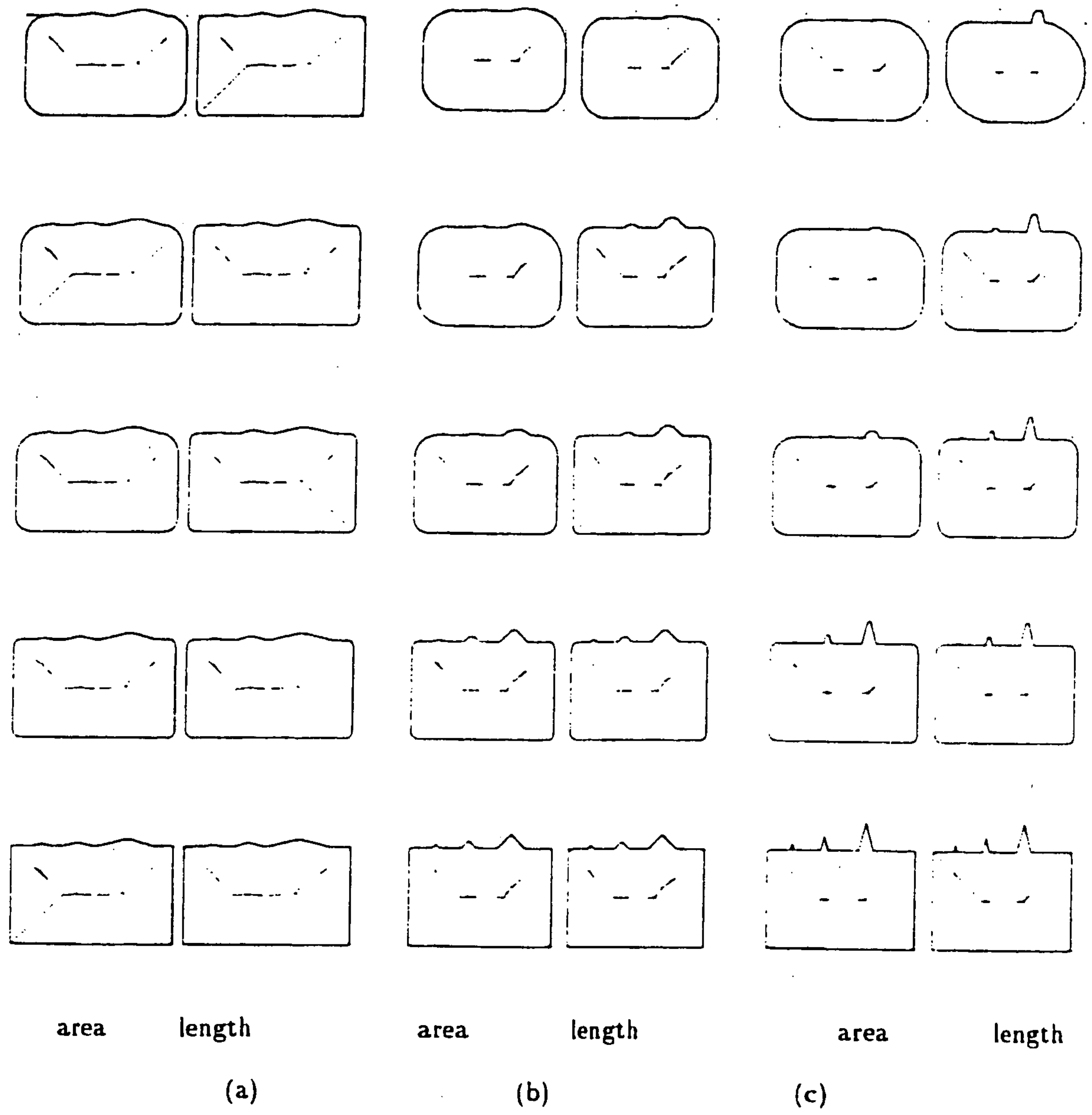


Figure 9.12: Salience measure of an axis point can be computed by from the area that is pruned by the prunning a axes point and the boundary length that is corresponding the prunned axis point. This figure depicts a pruning process operating on the saliency measures computed by the area (left) and the boundary length (left) of three different shapes. This figure illustrates that length by itself is not a good measure and area exhibits more stable results. In general, saliency measure based on the boundary length may be very sensitive to the addition of noise to local portions of shape, although there is little change to the shape's mass.

line segments. Then the area at a point s_i

$$A(s_i) = A(s_0) + \sum_{j=s_1}^{j=s_i} \Delta A(j) \tag{9.20}$$

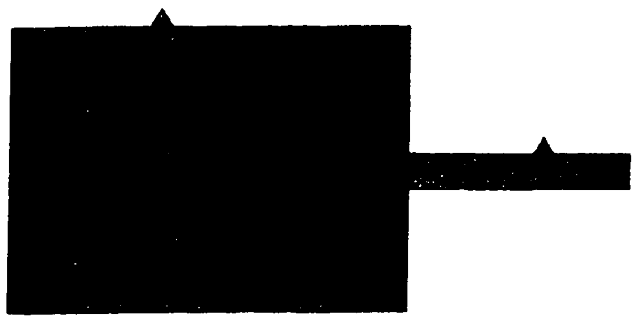


Figure 9.13: The significance of shape structure is dependent on its location in a given shape. In this figure, two protrusions of same size are added to two different parts of the shape. Observe that the one on the thin part is more significant than the other.

where s_0 corresponds to the first node of the symmetry axis that is in consideration. If s_0 corresponds to I node, the area, $A(s_0)$ is computed from the curvature of the boundary segment that forms s_0. If s_0 is a second order shock then $A(s_0) = 0$. If s_0 is a junction then the area, $A(s_0)$ is the sum of the values brought by each symmetry branch that terminates at this junction.

The area by itself as a saliency measure may lead to erroneous results. Specifically, for a given shape what is insignificant within one part of object may be significant in other part of the object. For example, Figure 9.13 illustrates a shape which has two same size protrusions placed in different places. It is clear that the protrusion placed in the thin part of the shape should be much more salient than the other protrusion. We propose to normalize the area by the radius of symmetry axis as a saliency measure:

$$SI(s_i) = \frac{A(s_i)}{R_i} \tag{9.21}$$

Thus, with relative area as the saliency function we are now able to make a distinction between what is important and what is not. In the next section, we present a pruning process based on these saliency measurements.

9.2.2 Pruning Processes

Assigning a saliency measure to each point in the symmetry axis is not sufficient to distinguish the globally salient features from the insignificant ones. A secondary process that removes a portion of the symmetry axes based on the saliency measurements and modifies the remaining symmetry set representation is required. The process that removes symmetry axes points (branches) is called *pruning*.

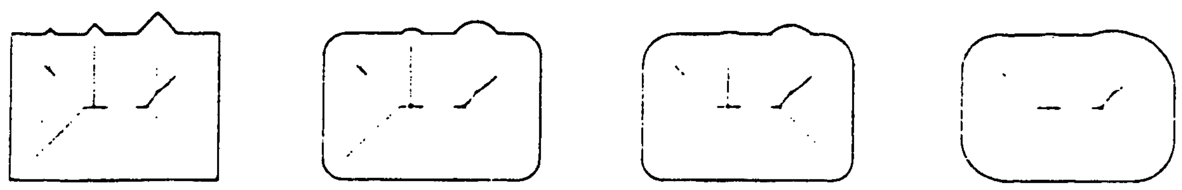


Figure 9.14: A shape is smoothed by the removal of axis points based on the saliency measure determined by the area function (9.20). Specifically, in each step a higher threshold value for saliency measure is applied to stop the smoothing. This approach unfortunately, cannot distinguish important features from insignificant ones. since smoothing is based on a local significance measure.

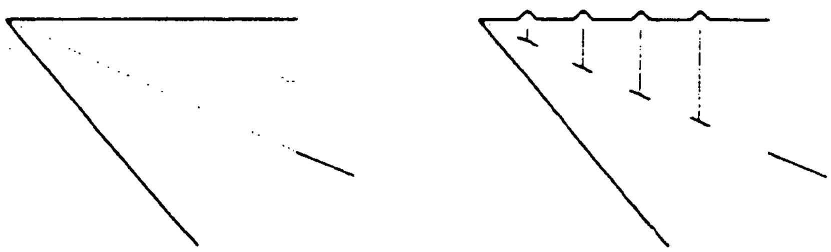


Figure 9.15: This figure depicts that globally salient symmetry axis can be broken into pieces due to the introduction of the symmetries from small protrusions. Each symmetry branch no longer represent a globally salient structure. A further grouping of the small segments into one axis segment is required.

Pruning processes can be classified into two groups based on how the symmetry axes are pruned: Some methods remove a *single point* from the symmetry axes. The simplest of them is based on the thresholding where all axis points whose salience measure is less than a certain threshold are removed. [24, 140, 58, 123, 149]. However, this technique may create disconnected axes if the saliency measure is not a increasing function of radius of symmetry axes. A similar but topology preserving approach is proposed by Shaked and Bruckstein [190] in which the pruning is performed on all axis end points in parallel. However, this approach smooths globally salient features while removing the axis points representing the small perturbations of boundary, Figure 9.14. The second set of methods operate on the symmetry branches instead of a single point. The first of these methods was observed by Blum and Nagel [25]. Ho and Dyer [83] have proposed a technique in

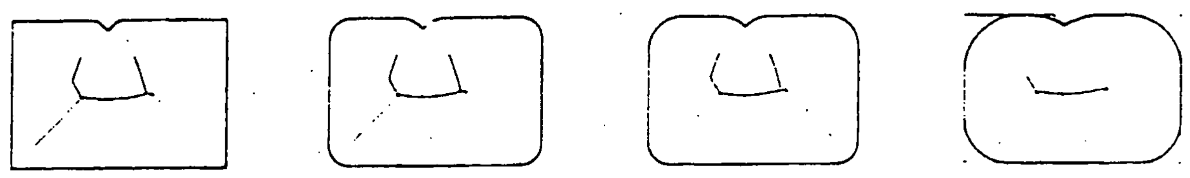


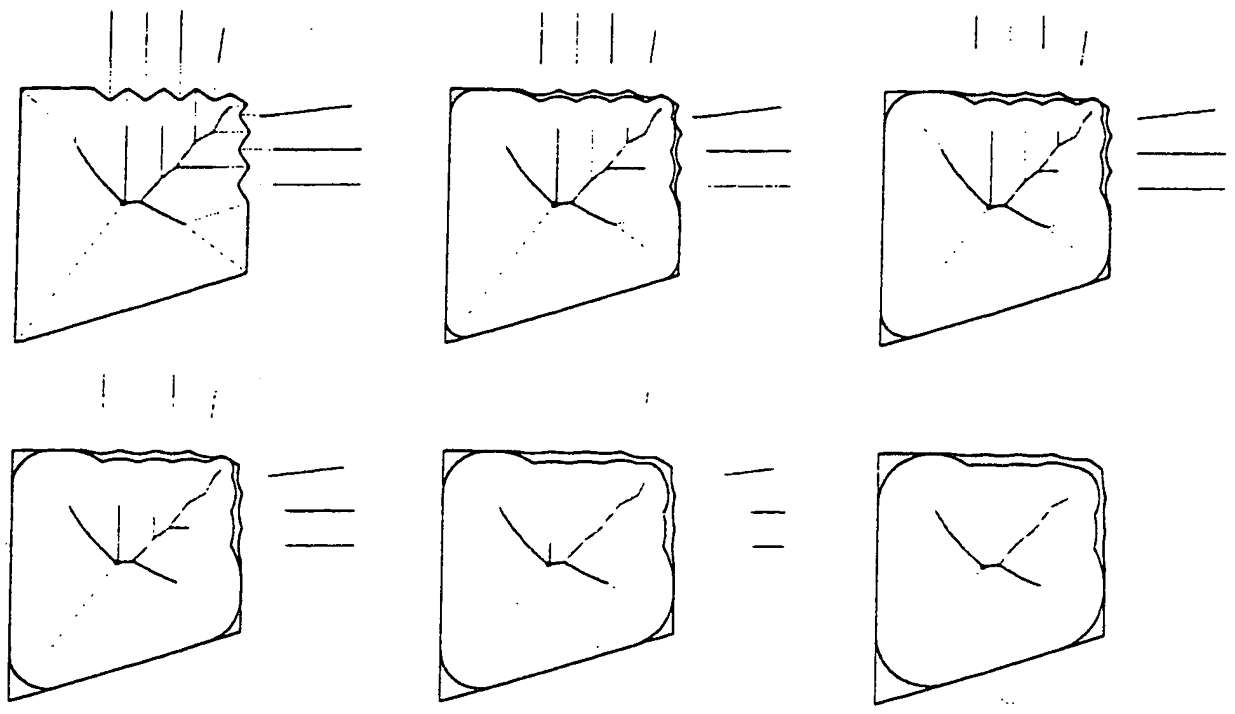
Figure 9.16: The addition of a small indentation to the shape boundary affects the inside symmetries drastically. Unfortunately, indentations cannot be smoothed by pruning the inside symmetries alone, thus requiring both inside and outside symmetry axes in pruning.

which an axis branch is removed if its saliency measure is less than a threshold. This is an interesting approach and brings more global shape information to the smoothing process. However, it cannot create a hierarchical scale space of a given shape since each symmetry axis *alone* does not necessarily correspond to a globally salient structure. In most cases, globally salient symmetry branches are broken into a lot of pieces and distorted around each junction due to the effect of less significant features, Figure 9.15.

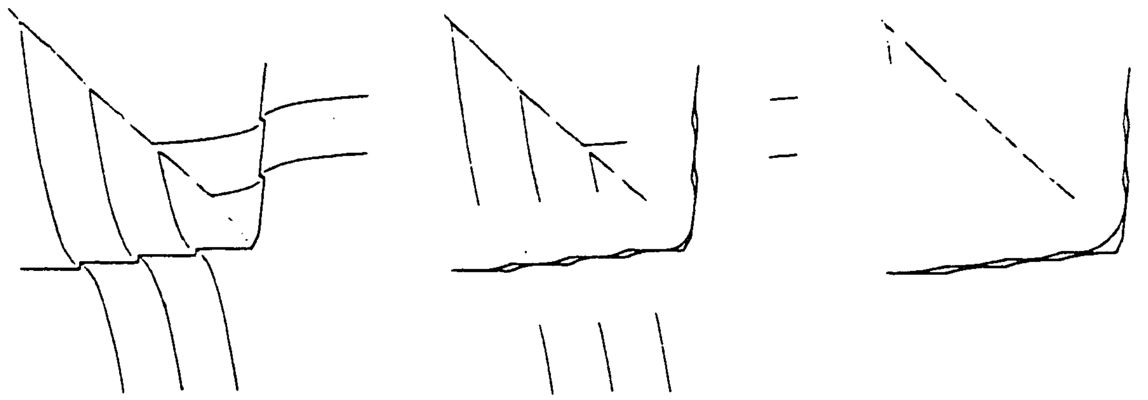
In the previous approaches (to our knowledge) *only* inside symmetries of a shape are considered in shape smoothing. However, each shape with a concave region has important outside symmetries as well. It may be impossible to smooth a concavity on a given shape boundary since concavities does not really form important symmetries inside the shape. Figure 9.16. An obvious approach to smooth these indentations is to consider outside symmetries in the smoothing process. Unfortunately, this does not yield a valid solution because smoothing the inside and outside symmetries creates *duplicate* boundary data. This problem is illustrated for a noisy rectangle, and for a noisy boundary segment. Figure 9.17.

9.2.3 A Coupled Symmetry/Boundary Smoothing Approach

In this thesis, we propose a coupled symmetry/boundary smoothing algorithm. Specifically, our approach consists of three stages: (*i*) removal of a symmetry axis, (*ii*) smoothing/grouping boundary segments due to the removal of an axis segment at (*i*). (*iii*) recomputing the symmetry set and the saliency function. The last two stages are necessary since modification of a part of the symmetry axes modifies the boundary and consequently the remaining symmetry axis. One may think that this is a computationally expensive algorithm due to the recursive nature of pruning and re-computation of symmetry axes.



(a)



(b)

Figure 9.17: The shape (or boundary) smoothing based on the prunning symmetry axes (both inside and outside) leads to the duplicate boundary formation. In each prunning process removal of an axis segment pulls a convex boundary segment inward (or outward in the case of outside symmetries), thus creating invalid boundary representation. (a) noisy quad and (b) noisy boundary segment c ntaining a corner.

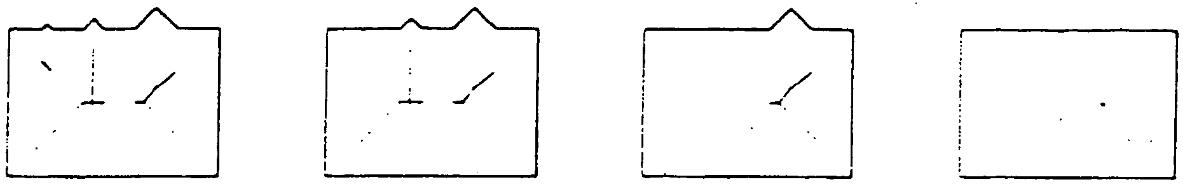


Figure 9.18: This figure illustrates a pruning process based on the removal of axis branches. In each iteration, the least salient branch is pruned and the saliency of the symmetry branches that it affects are recomputed. This process places different object features onto scale space.

However, these computations are all done locally by canonical symmetry operators and thus very efficient.

The first step of our algorithm at each iteration is the removal of the least significant symmetry axis branch. This operation corresponds to deforming the shape to a simpler one by choosing the *least action* path to it. We believe that pruning should operate on the symmetry axis branches instead of axis points. This intuition is based on the observation that each branch corresponds to a feature for a given object [25, 57, 160]. For example, consider the example illustrated in Figure 9.14 where pruning based on the axis points cannot distinguish the important features from the non-important ones, as the axis points do not contain the necessary global information about the shape. Alternatively, an iterative branch removal process illustrated in Figure 9.18 places different object features into scale space.

The second step of our algorithm is the replacement of the boundary segment due to the removal of a symmetry axis branch. Consider the boundary segment containing a step edge, Figure 9.19. The removal of an axis branch means that the boundary between points A and B should be smoothed as such that the symmetry branch must not form. This accomplished by constructing the smooth boundary using a circular arc so that no symmetry axis will form in the shaded area, Figure 9.19. Biarcs or Euler-spiral [102] can also be used to approximate the smooth boundary segment.

The last step of our algorithm is the reconstruction of symmetry axes and their saliency. This step is necessary because parts of the symmetry axes are no longer valid due to the introduction of new boundary segments in the previous step. For example, symmetry axis, SA_2 and its saliency measure in Figure 9.19 must be modified. However, the re-computation

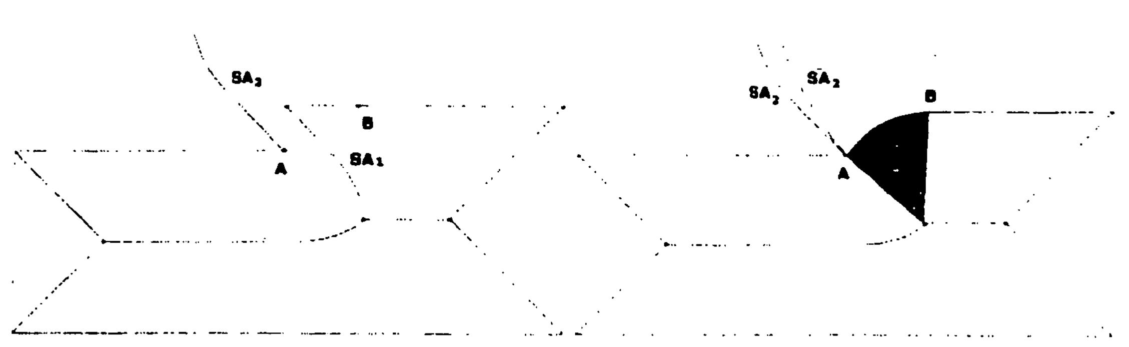


Figure 9.19: This figure illustrates the coupled symmetry/boundary smoothing algorithm. First, the symmetry axis, SA_1 is deleted from the symmetry map. This requires that the boundary between A and B be modified so that no shocks (or symmetries) form in the influence zone of boundary segment AB, (shaded region). This is accomplished by replacing the boundary segment with a smooth boundary segment, *e.g.*, circular arc. Observe that when the boundary segment AB is smoothed by a circular arc, the symmetry axis SA_2 is no longer valid. Thus, the re-computation of symmetry axes and their saliency is axes required. $\overline{SA_2}$ is the correct symmetry branch.

of symmetry set can be computationally expensive if special care is not taken, thus making the smoothing algorithm impractical. In addition, the accuracy of the symmetry detection is crucial as such the errors due to the inaccurate detection of symmetry may accumulate during this iterative symmetry/boundary smoothing process. This three stage coupled symmetry/boundary smoothing algorithm corresponds to the splice transform introduced in the previous chapter.

Recall that symmetry transforms can be cast as a combination of deletion and addition operators. The splice transform, for example, effectively replaces the boundary segment corresponding to a shock branch by a circular arc. Note that this is different from the thresholding on symmetry axis points, as proposed in [190]. Entire branches, not points, are affected because the removal of a branch affects the remaining symmetries, *i.e.*, on the other side of the contour and in the vicinity of the symmetry branch. Figure 9.20 and Figure 9.21.

Figure 9.22 illustrates the effect of a splice transform in conjunction with a gap/part transform. Observe that the existence of numerous negative curvature minima leads to numerous part hypotheses. The smoothing of small scale bumps while retaining the coarse-scale "corners" was a major difficulty in a previously proposed partitioning approach [195]. The iterative sequence of splice transforms, guided by distance to transition, recovers this coarse scale structure to the point where parts are evident, and the application of a gap/part transform can easily recover the part structure.

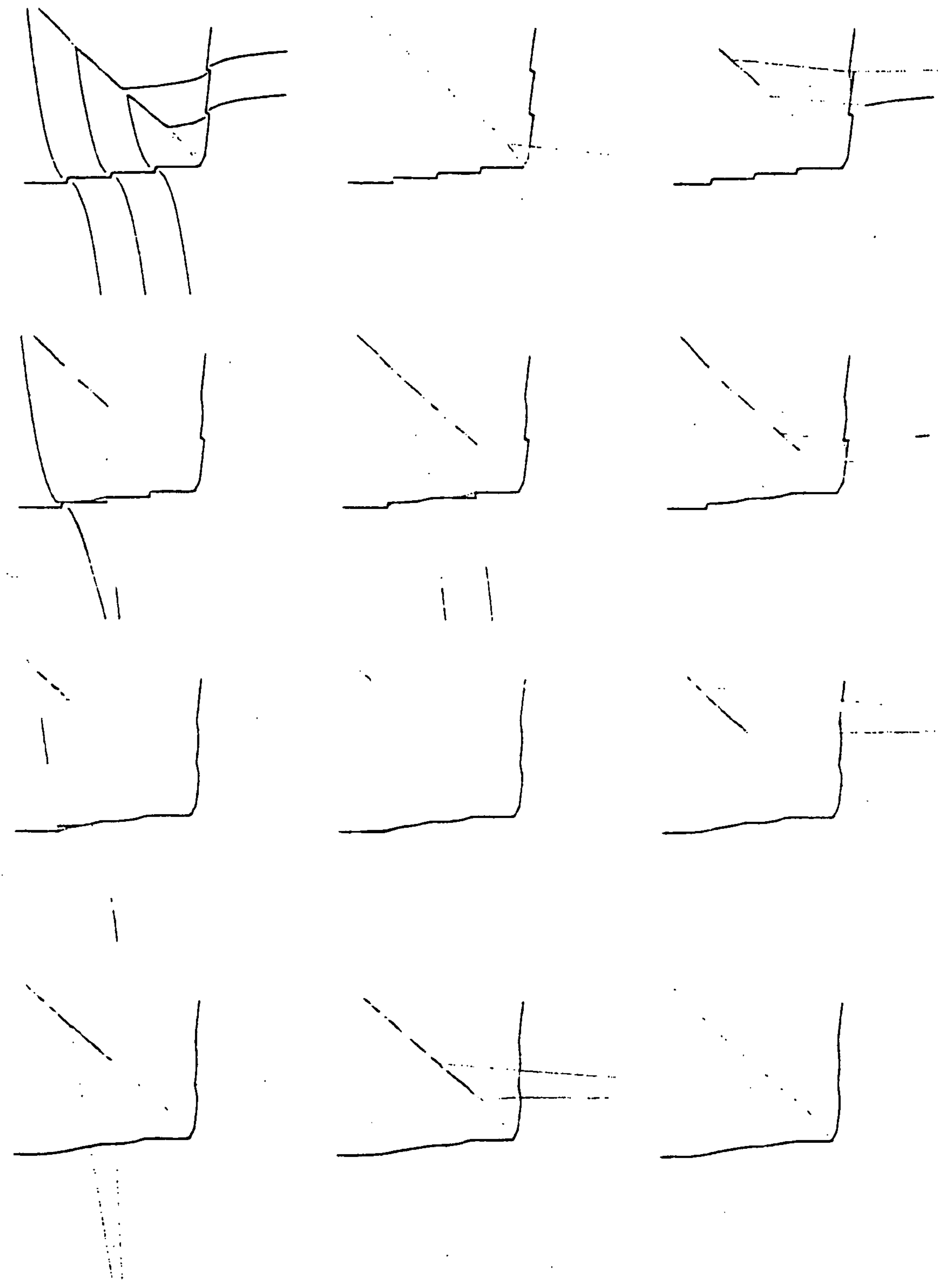


Figure 9.20: This figure illustrates that an iterative shock branch removal technique leads to a boundary smoothing. Note that the corner is not smoothed and boundary is grouped as broken into two piece which is represented by piecewise circular arcs.

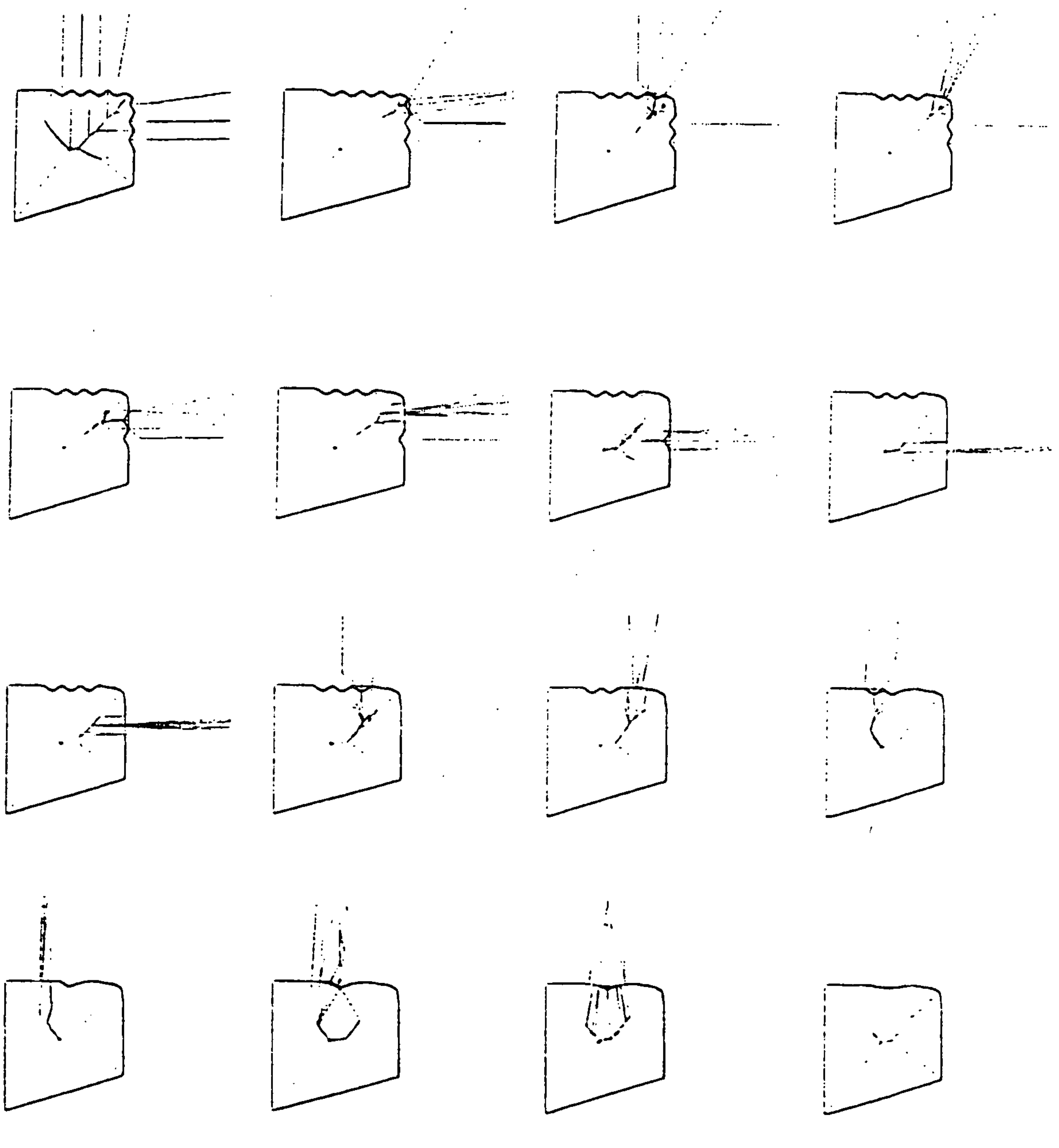


Figure 9.21: This figure illustrates that an iterative shock branch removal technique leads to a shape smoothing. Note that the corners are not smoothed and boundary is grouped as broken into four piece which is represented by piecewise circular arcs.

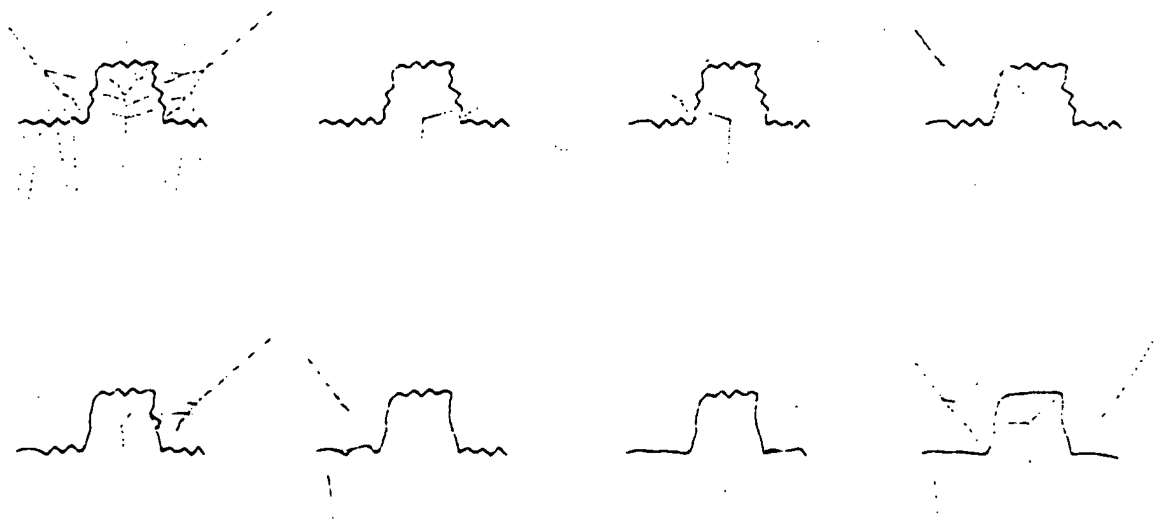


Figure 9.22: This figure illustrates that the iterative application of a splice transform removes shock branches leading to boundary smoothing. Gap/part transform. Figure 8.10 is now applicable.

Part III

Shock-Based Reaction-Diffusion Bubbles For Image Segmentation

Chapter 10

Introduction

Figure-ground segmentation is a fundamental problem in computer vision and presents a bottleneck to the recovery of objects from images. Previous approaches to this problem are either "bottom-up" in that low-level which are pixel-based features such as edges are integrated in the form of progressively more global contours, or "top-down" in that high level models are verified on the image [169, 19, 244]. The bottleneck exists principally because it is the interface between low-level features which are image based, and high-level descriptions which are object based. An approach to resolving this difficulty is to present an intuitive intermediate language which can be computed from low-level features, but which can also be influenced by high-level function.

In Chapter 8 and 9, we proposed that a symmetry map can be used explicitly as an intermediate representation between low-level edge maps and the high-level object boundaries. Specifically, all grouping operations such as completing gaps, pruning spurious elements as well as smoothing and partitioning shape are expressible as a sequence of symmetry transforms on the initial symmetry map. The optimal grouping corresponds to the least action path.

Previously, active contours [95, 223, 222], or snakes, were originally proposed as such an intermediate representation to provide "alternate organizations among which high-level processes may choose" [95]. Other physically motivated deformable models extend this framework to address its initialization limitations by the use of an "inflation force" [223, 48], and its topological difficulties by using a level set approach [34, 134, 212, 213, 98, 35] and also by reparametrization [208]. While the basic idea underlying these models is to *automate* aspects of segmentation, much interactive attention is still required, specifically, to initialize

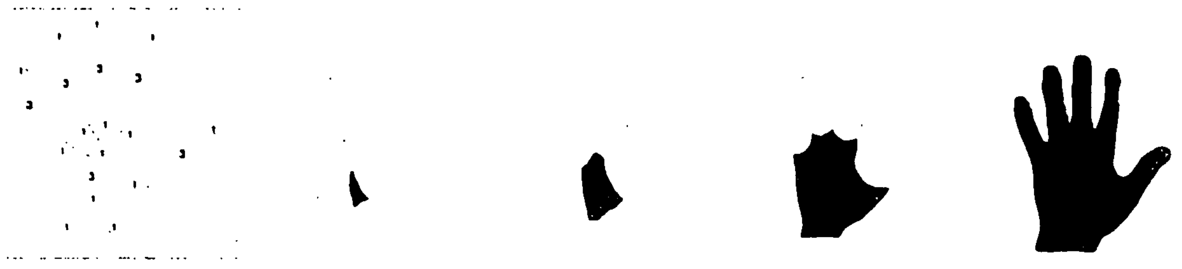


Figure 10.1: The shock-based description of the image of a hand, and its growth from shocks. This reconstruction is effectively the union of disks centered on shocks, thus requiring all the shocks [196].

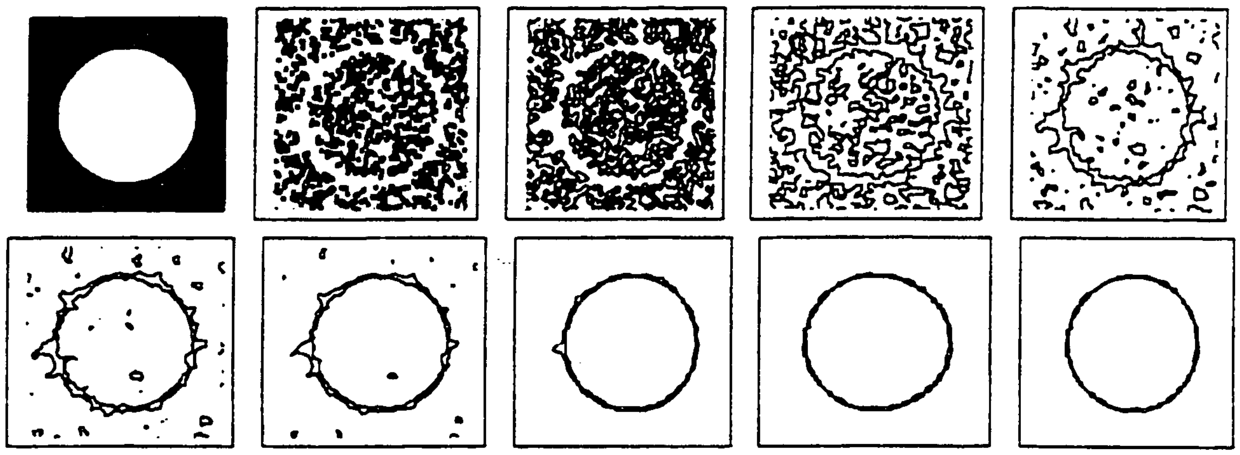


Figure 10.2: Bubbles are first randomly initialized in the homogeneous areas of the image, grow and merge to form larger bubbles, or split into smaller bubbles and deform to adhere to object boundaries under the influence of gradients (*e.g.* edges) in the image. Finally, the sequence converges to two bubbles trapping the object boundaries, one from the inside and one from the outside.

the deformable models appropriately: one deformable model must be initialized per object of interest such that it is near and symmetric to the object's boundary. Furthermore, some deformable models do not handle gaps in the low-level edge maps and narrow regions present difficulties to others.

An alternative approach that resolves some of these difficulties constructs a representation for a shape *directly* from the image information. A complete representation of a given shape in the "shape from deformation" framework [106, 112] is obtained by detecting, classifying and grouping four types of *shocks* which are formed when the shape is evolved by a *reaction-diffusion* process. The reverse-time reconstruction of shape from shocks yields a shock-based morphogenetic language which views shape dynamically: fourth-order shocks (seeds) are born and grow to join other parts at second order shocks; first-order shocks represent boundary deformation while third-order shocks represent regional deformations, Figure 10.1. This is the description of a *known* shape which has already been segmented from the image. However, how can this representation be obtained *prior* to figure-ground segmentation? We propose to hypothesize simple object descriptions (bubbles) which are initialized and then evolved consistently with the image information to recover a complete description of the underlying shapes. Specifically, we hypothesize *randomly-placed* fourth-order shocks to capture unknown objects and their parts. These fourth-order shocks grow, merge, split, shrink, and disappear, and in general deform unless influenced by image information to confirm or invalidate the existence of an object and its parts, Figure 10.2. Thus, initial regional homogeneities interact and combine under the constraint of boundary gradients to segment figure from ground.

A *duality* between boundaries shocks is implicit in the above discussion. Figure 10.1 shows how waves generated at inner shocks generate the shape completely. The same is true of outer shocks if the shape is embedded in a compact domain with boundaries, *e.g.*, image domain instead of R^2. Therefore, waves generated at inner and outer shocks can generate the shape, and therefore simultaneously arrive and quench at the shape boundary. Hence, the shocks resulting from these waves is the original boundary, *i.e.* shocks of shocks (when considered as boundaries) yield the original boundary. This *duality* between *shocks* and boundaries connects the two approaches.

This shock-based reaction-diffusion bubbles technique works well when object boundaries have moderate intensity gradients, even when small gaps are present [92]. However, the *convergence* of bubbles remains unresolved under certain conditions involving weak or diffuse boundaries, large gaps, edges homogeneous only one side, and very narrow regions. This is due to the monotonic nature of growth: once a region has evolved beyond object boundaries it can no longer return to capture it. In Chapter 13, we proposed the *skeletally coupled deformable models* to deal with the convergence issue of the bubbles technique [178]. Specifically, initialized seeds grow in a curve evolution implementation of active contours, but where growth is modulated by a skeletally-mediated competition between neighboring regions, thus combining the advantages of local and global region growing methods. This approach effectively deals with many of the difficulties presented above as illustrated by numerous examples in [178].

Several interesting relevant approaches have appeared after this work was originally reported [212, 213]. Kichenassamy *et al.* [98] and Casalles *et al.* [35] independently considered the geometric level set evolution in the context of "snakes" based on energy minimization. They showed that the new model is more robust to initialization and guarantee convergence the local minimum. Similarly, Whitaker [232] recently considered the original 2D and 3D snakes model in a PDE form and proposed a fast numerical method for these implicit methods. Shah [189] presented a common framework for curve evolution, segmentation and anisotropic diffusion.

In this part, we also generalize the 2D reaction-diffusion bubble technique to segmenting three-dimensional images [213, 211, 214]. We consider three approaches. First, we examine the segmentation of each slice of the image along some axis and the re-assembly of the results. We will show that in addition to other difficulties, a large gap in one slice will often prevent successful segmentation. As such the segmentation of slices along different axes may lead to different results. To deal with large gaps in one slice, information about differential structure can be diffused to neighboring slices. In this $2\frac{1}{2}$D approach, 2D bubbles are guided by 3D information, *e.g.*, 3D intensity gradients. However, this approach is successful for only shallow depth large gaps, and even then at the expense of blurring across discontinuities. Finally, we employ three-dimensional bubbles in a reaction-diffusion space. While the reaction process in three-dimensions is trivially extended, the generalization of diffusion is not straightforward. We utilize a particular Mean-Gauss curvature deformation to serve as the diffusion process. The three-dimensional reaction-diffusion bubbles are intrinsic, can deal with a variety of gaps, and place captured surfaces in a hierarchy of scale.

Two major issues arise in the segmentation of 3D images by the 3D bubbles. approach. First, in analogy to 3D balloons [48], 3D images may be segmented by (i) an independent segmentation of each 2D slice: (ii) by segmenting each 2D slice with the aid of nearby slices; or (iii) by segmenting the 3D image by 3D bubbles. We examine each approach with respect to accuracy of reconstruction, efficiency, stability of segmentation to changes in the slice plane, and robustness of results in cases when gaps are present and conclude that a treatment in 3D is most appropriate, Section 12.1. This raises a second issue on how the bubble deformation may be regularized to bridge 3D gaps. In 2D, the addition of a curvature-driven deformation leads to "stiffer" bubbles which do not penetrate gaps. A generalization of this process to 3D produces a Mean-Gaussian curvature-dependent surface flow which regularizes the flow of 3D bubbles in presence of 3D gaps, Section 12.2.

Chapter 11

2D Reaction-Diffusion Bubbles

This chapter is organized as follows. In Section 11.1 we review active contours and related research. In Section 11.2 we present the reaction-diffusion bubble technique and illustrate it on synthetic and medical images. In Section 11.3 we present implementation issues, several examples, and some difficulties associated with the bubble method.

11.1 Active Contours

In this section we briefly review the "active contour" literature as is pertinent to this paper. Active contour, or snakes, were initially proposed as energy minimizing splines [95, 223, 222], and were modified and applied to realistic images [124, 125, 8, 67, 54, 172, 22]. Two excellent approaches were proposed to improve the initial framework, which we will review in detail as our model is inspired by, and is based on them. The first approach is the *balloon* technique of Cohen and Cohen [46, 47, 48], while the second approach is the *level set approach* of Caselles *et al.* [33] and Malladi *et al.* [133, 134]. Other deformable models have also been proposed: an interesting probabilistic model, the "ripple filter", was earlier proposed by Cooper *et al.* [49]; see also [239, 205, 39, 18].

11.1.1 Snakes

Active contours, or snakes [95], are deformable models based on energy minimization of controlled-continuity splines. When they are placed near the boundary of objects they will lock onto salient image features under the guidance of internal and external forces. Formally, let $C(s) = (x(s), y(s))$ be the coordinates of a point on the snake, where s is the length parameter. The energy functional of a snake is defined as

$$E(\mathcal{C}) = \int_0^1 [E_{int}(\mathcal{C}(s)) + E_{image}(\mathcal{C}(s)) + E_{con}(\mathcal{C}(s))]\, ds. \tag{11.1}$$

where E_{int} represents the internal energy of the spline due to bending, E_{image} represents image forces, and E_{con} are the external constraint forces. First, the internal energy,

$$E_{int} = w_1|\mathcal{C}'(s)|^2 + w_2|\mathcal{C}''(s)|^2. \tag{11.2}$$

imposes regularity on the curve. and. w_1 and w_2 corresponds to elasticity and rigidity. respectively.Second, the image forces are responsible for pushing the snake towards salient image features. Kass *et al.* considered image intensity for lines. intensity gradient for edges. and the curvature of level lines in a slightly smoothed image for terminations. They pointed out that a combinations of these image forces can create a wide range of snake behavior. Third, the external constraint forces. which come from user interface. or high-level interpretations, are to put the snake in the vicinity of the desired local minimum. The local behavior of a snake can be studied by considering the Euler-Lagrange equation.

$$\begin{cases} -(w_1\mathcal{C}')' + (w_2\mathcal{C}'')'' = F(\mathcal{C}). \\ \mathcal{C}(0), \mathcal{C}'(0), \mathcal{C}(1) \text{ and } \mathcal{C}'(1) \text{ given}. \end{cases} \tag{11.3}$$

where $F(\mathcal{C})$ captures the image and external constraint forces. Note that the energy surface E is typically not convex and can have several local minima. Therefore, to reach the solution closest to the initialized snake, the associated dynamic problem is solved instead of the static problem [95, 46, 47, 48, 125]. When the solution $\mathcal{C}(t)$ stabilizes. a solution to the static problem is achieved.

$$\begin{cases} \frac{\partial \mathcal{C}}{\partial t} - (w_1\mathcal{C}')' + (w_2\mathcal{C}'')'' = F(\mathcal{C}) \\ initial + boundary\, conditions \end{cases} \tag{11.4}$$

Snakes perform well when they are placed close to the desired shapes. In general, however: *i*) if a snake is not close to an edge, it is not attracted by it and instead. in the absence of other image forces, deforms to a line or a point. Figure 11.1; *ii*) snakes might display oscillatory behavior because of high intensity gradients which are used to push the snakes towards edges; *iii*) the original numerical solution to energy minimization might also lead to unnecessary oscillations; *iv*) the selection of the elasticity and rigidity parameters, which play an important role in the behavior of snakes, is ad-hoc.

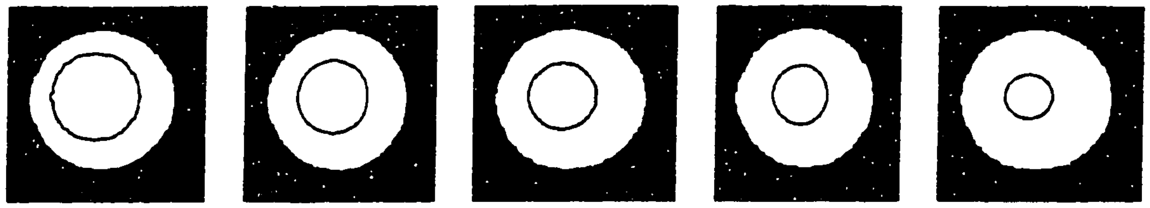


Figure 11.1: When a snake is not placed in the vicinity of the image features, it is not attracted by them and shrinks away from the object of interest.

Several approaches have been suggested to deal with these difficulties. The stability of snakes has been investigated by adjusting internal parameters in [67, 172, 22]. Leymarie and Levine [125] introduced bounds on the image force, new rules for setting the elasticity parameters, and a new terminating condition. They used snakes to represent the shape of planar objects using the distance transform to minimize the energy function [124], and to track the nonrigid objects, *e.g.* cells [125]. Amini *et al.* [8] used dynamic programming to improve stability. Neunschwande *et al.* [146] presented a method in which a user only specifies the endpoints of the initial curve. David and Zucker integrated local edge evidence obtained by a relaxation labeling procedure [87, 245] by using snakes which slide down potential surfaces constructed from these edge labels [54]. Berger and Mohr [22] also used several small growing snakes to detect local behaviors. Cooper *et al.* [49] presented a probabilistic model called a "ripple filter", which acts as a deformable contour, see also [239, 205, 39, 18] for other deformable models. Despite these improvements, a number of fundamental difficulties remain: in particular, snakes heavily rely on a proper initialization close to the boundary, multiple initializations, one per object of interest, *etc.* We now review the balloon and the level set approaches which discuss these issues.

11.1.2 Balloons

To overcome some of the initialization difficulties with snakes, Cohen and Cohen [46, 47, 48] introduced a deformable model based on the snakes idea. This models resembles a "balloon" which is inflated by an additional force which pushes the active contour to object boundaries, even when it is initialized far from the initial boundary. The image forces, *e.g.* $P(C) = -|\nabla I(C)|^2$, are normalized to avoid instabilities, otherwise the curve can be trapped by spurious isolated edge points. Cohen *et al.* also take into account edge points previously detected by a local edge operator to get a better performance, leading to an improved external force

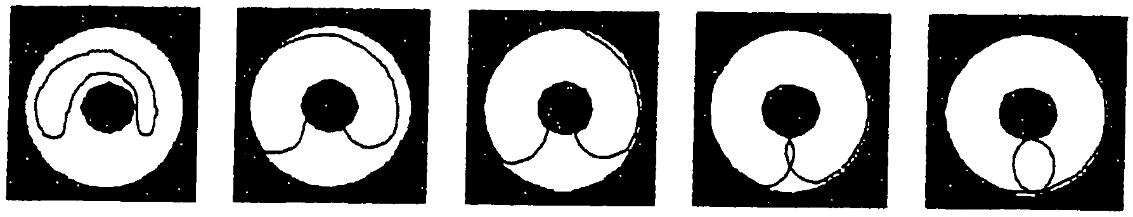


Figure 11.2: This figure illustrates that balloons cannot handle the topological changes such as merging, and splitting.

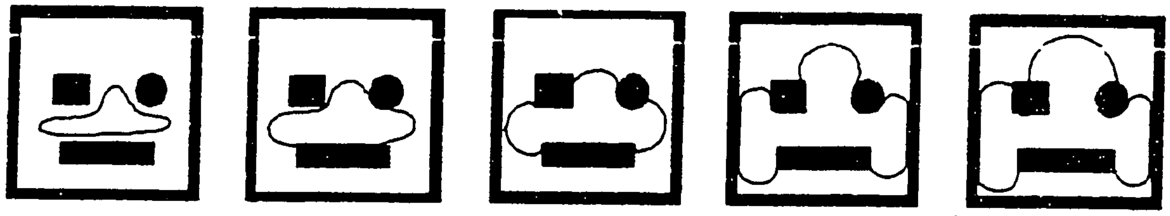


Figure 11.3: A balloon which is initialized on a synthetic image containing three objects. The evolution of the balloon is from left to right up to the time of convergence. Since the balloon cannot split into two balloons, this figure illustrates that for every object in the image the user has to initialize a new balloon. However, this leads to excessive user interaction in the case of multiple objects in a given image.

$$F = c_1.\bar{N} - c\frac{\bar{\nabla}P}{||\bar{\nabla}P||}, \tag{11.5}$$

where c_1 is the inflation force, $\bar{N}$ is the unit normal vector, and c is a constant. Typically, c is selected slightly larger than c_1 so that an edge segment can stop the inflation force. Therefore, the basic idea is that the balloon expands, passes over weak edges, and stops at the strong edges.

Despite these improvements on snakes, balloons still have some problems: First, like snakes, balloons cannot handle *topological changes, i.e.*, merging and splitting. Figure 11.2 illustrates that when two moving fronts hit each other they do not merge, but rather pass over each other. Furthermore, a single balloon cannot capture more than one object. Figure 11.3. Therefore, when there are multiple objects in the image, further user interaction is required. Second, Figure 11.4 illustrates that a balloon cannot capture objects with sharp protrusions because of the way internal parameters are chosen. Since w_1 and w_2 are chosen to be order of h^2 and h^4 [46, 47, 48] to obtain typically good results, sharp protrusions and indentations cannot be captured (however, adjusting these parameters *can* produce a wide range of balloon behavior). Third, the selection of the inflation force term, which is used to

11.1.3 Level Set Evoluti n

One of the traditional problems of snakes and balloons is that they cannot easily capture topological changes. Therefore, for images with multiple objects, the snake or balloon methods require extensive user interaction. This problem can be resolved by the use of the level set evolution, proposed by Osher and Sethian for flame propagation [154, 183, 20], introduced to computer vision in [106, 112], and first applied to active contours in [33, 133].

The level set approach consider a curve C as the zero level set of a surface, $\phi(x, y) = 0$. Caselles *et al.* [33] proposed that the the zero level set of the function ϕ, $\{x \in R^2 : \phi(t, x) = 0\}$, evolve in the normal direction according to

$$\frac{\partial \phi}{\partial t} = g(x, y)|\nabla\phi|(div(\frac{\nabla\phi}{|\nabla\phi|}) + v), \tag{11.6}$$

where $g(x, y) = \frac{1}{1+(\nabla G_\sigma * I)^2}$, v is a positive real constant, $G_\sigma * I$ is the convolution of the image I with the Gaussian G_σ, and, ϕ_0 is the initial data which is a smoothed version of the function $1 - X_\Gamma$, where X_Γ is the characteristic function of a set Γ containing the object of interest in the image. The gradient of the surface $\nabla\phi$ is the normal to the level set C, $\vec{N}$, and the term $div(\frac{\nabla\phi}{|\nabla\phi|})$ is its curvature κ, so that equation (11.6) written as

$$\phi_t - g(x, y)(v + \kappa)|\nabla\phi| = 0, \tag{11.7}$$

can also be viewed as a modified reaction-diffusion curve evolution $C_t = g(\beta_0 - \beta_1\kappa)\vec{N}$. Caselles *et al.* [33] proved the existence and uniqueness of solutions of this PDE in the viscosity sense [128] for bounded Lipschitz continuous initial data.

Malladi *et al.* [133, 134] independently proposed the following equation,

$$\phi_t + S(x, y)(\beta_0 - \beta_1\kappa(x, y))|\nabla\phi| = 0, \tag{11.8}$$

where $S(x, y)$ is the image based speed function defined for all level sets, extended from a speed function on the zero level set defined as

$$\hat{S}(x, y) = \frac{1}{1 + |\nabla G_\sigma * I(x, y)|}. \tag{11.9}$$

The image-based speed function has values that are close to zero in regions of high intensity gradient and values that are close to unity in regions of constant intensity. This function defined for the zero level set is expanded to the other level sets which prevents the level sets from colliding with each other.

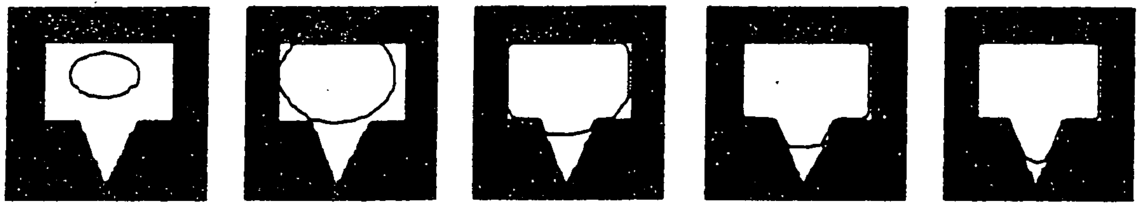
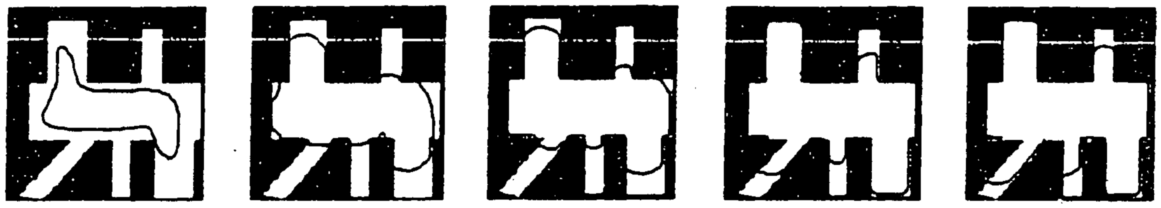
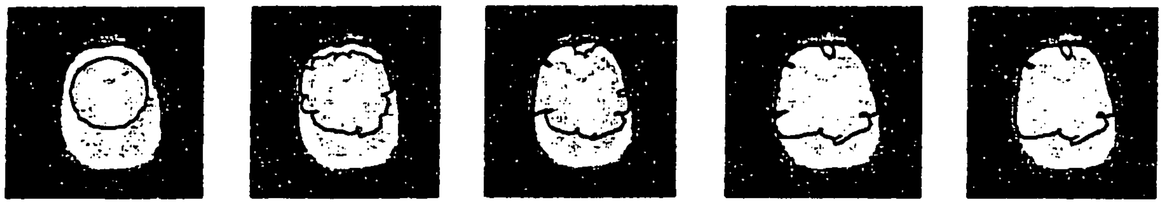


Figure 11.4: A balloon cannot capture sharp protrusions or indentations for a typical selection of elasticity and rigidity parameters.



a)



b)

Figure 11.5: This figure illustrates an inherent difficulty associated with the choice of inflation force. The sequence begins with the left image and converges at the right image. (a) The inflation force pushes the balloon into narrow regions. but also over the boundary at other places. (b) A low inflation force stops the balloon at isolated and noisy edges, as illustrated on an MRI brain image.

push the balloon to object boundaries and plays an important role, is ad-hoc. The problem is an inherent trade-off in the choice of a magnitude for the inflation force: on the one hand. a high inflation force might cause the balloon to cross over the object boundaries, especially when it displays oscillatory behavior: on the other hand. a low inflation force will not be sufficient for the balloon to pass over weak edges or noise, Figure 11.5. Finally, the balloon method is semi-automatic in that the user is required to initialize a balloon properly for each object in the image. Some of these issues are resolved by the level set approach which is described next.

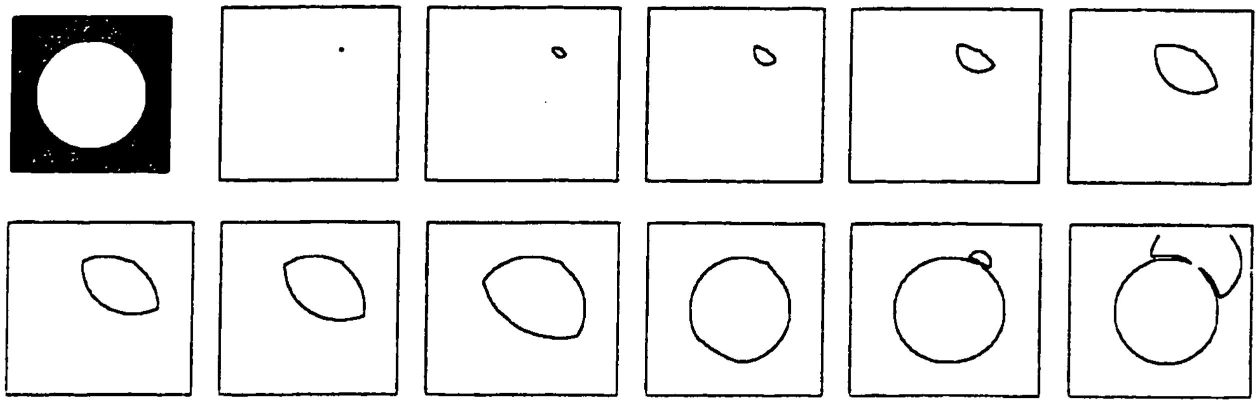
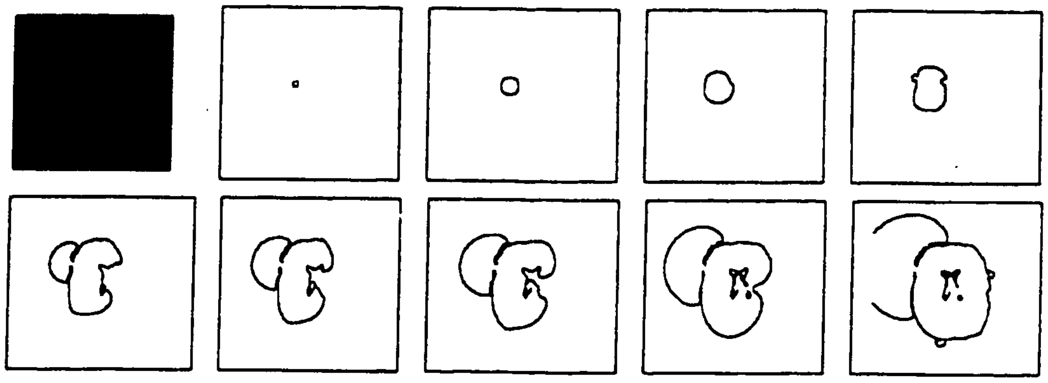


Figure 11.6: The speed function $S(x, y)$ is not stationary on the boundaries. If the initial contour is placed close to portion of the object boundary, the active contour crosses over the closer portion while it is approaching the distant portion.

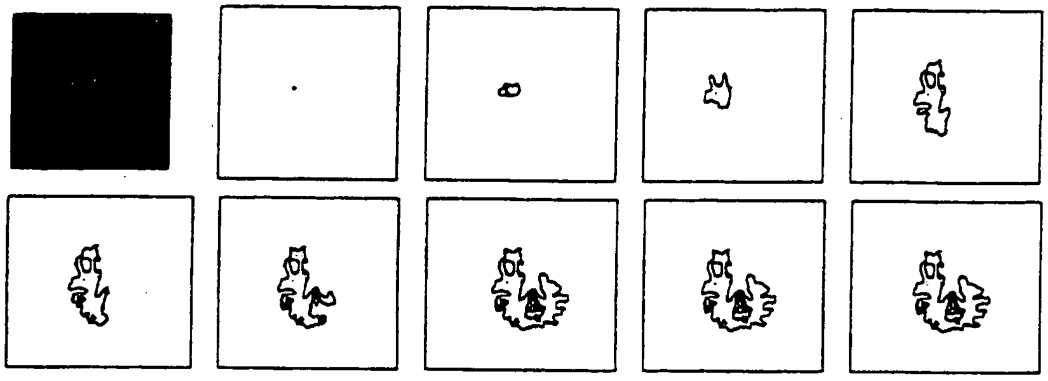
Unlike snakes, this active contour model is intrinsic, stable, *i.e.*. the PDE satisfies the maximum principle, and can handle topological changes such as merging and splitting without any computational difficulty [33, 133, 134]. They can be used to extract smooth shapes, and to find several contours simultaneously if they are initialized properly. Despite these great improvements, several difficulties remain which are discussed below:

1. Symmetric initialization: This approach works well if the initial contour is placed nearly symmetrically with respect to the boundaries of interest, so that level set reaches object boundaries almost at the same time. However, if the initial contour is much closer to one portion of object boundary than another, Figure 11.6, the evolving contour crosses over object boundaries closest to it. This is due to the fact that the speed function is small, but not zero, near the boundaries, causing the active contour to continuously deform even if it is a small amount. Therefore, while the evolving contour is approaching the distant boundary, it can cross over the nearer portion, as illustrated in Figure 11.6.

One possible solution is to make boundaries much stronger by choosing a larger m in the speed function $\frac{1}{1+|\nabla G_\sigma * I(x,y)|^m}$ (recall that Caselles *et al.* used $m = 2$ while Malladi *et al.* used $m = 1$). However, this introduces the following problems. First, the level set will converge slightly away from the boundary. Second, the narrow regions will become inaccessible to the level sets. Third, any noise in the image will have a strong effect on the results since it would be amplified. An alternative solution is to threshold the speed function. However, this introduces the following trade-off in the selection of a threshold: on the one hand, if a high threshold is used, level sets will pass over weak boundaries,



(a)

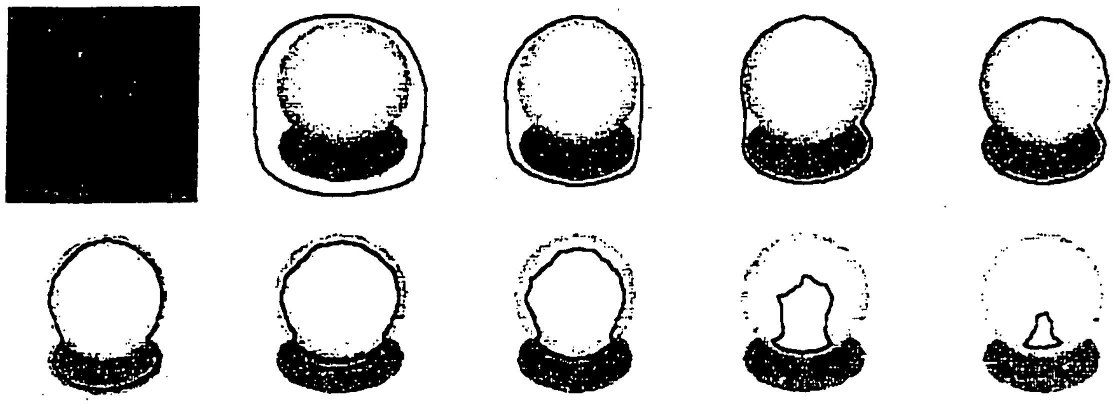


(b)

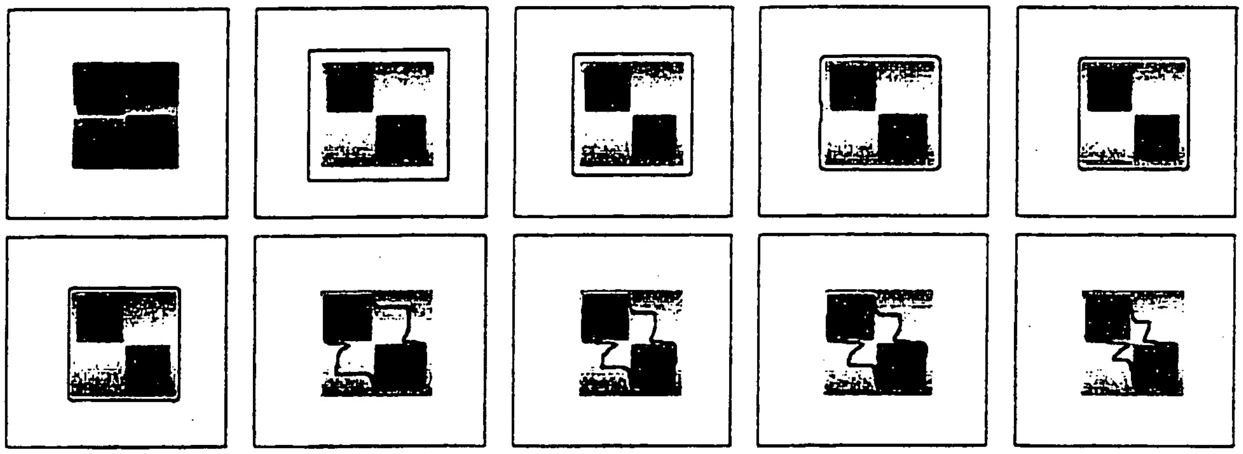
Figure 11.7: This figure illustrates inherent difficulties with thresholding the speed function. as a possible solution to the "symmetric initialization" problem. (a) A high threshold will cause the level set to cross over weak boundaries. (b) A low threshold will cause the level set to stop at noisy or isolated edges.

Figure 11.7a; on the other hand. a low threshold stops the evolving contour at the isolated or noisy edges, Figure 11.7b. leading to possibly undesirable results.

2. Multiple initialization: A single level set may not capture the object of interests if they are embedded in other objects. Indeed, Malladi *et al.* [133] recognize this point and suggest a two-stage algorithm for capturing the intermediate objects. In this approach. in the first stage, the level set captures the outer boundary, first followed by a second stage where this level set becomes the initialization for a second evolution to capture the intermediate objects. While this works in some cases, in images containing line, or asymmetrically situated embedded objects, this method (or a multistage extension) will not capture the underlying objects. Figure 11.8a illustrates a synthetic sphere with an attached shadow



(a)



(b)

Figure 11.8: This figure illustrates problems associated with a single level set initialization. (a) This figure depicts a two stage algorithm intended to capture synthetic sphere and its attached shadow. The top row is the first stage evolution whose end result is then used for the second stage, bottom row, by "momentarily relaxing the image-based speed function" [133]. This example is representative of objects with interval lines connecting its boundary, as well as objects situated very close to each other. (b) A synthetic image depicting two binary squares embedded in a gray level square region. The top row depicts the first stage of evolution whose result is then used as the initial contour for the second stage, bottom row. As is evident, due to the *nearly symmetric initialization constraint* the volving contours which is initially close to the black squares cross over its boundary at closer portions while its distant portions are evolving towards the central part of the image.

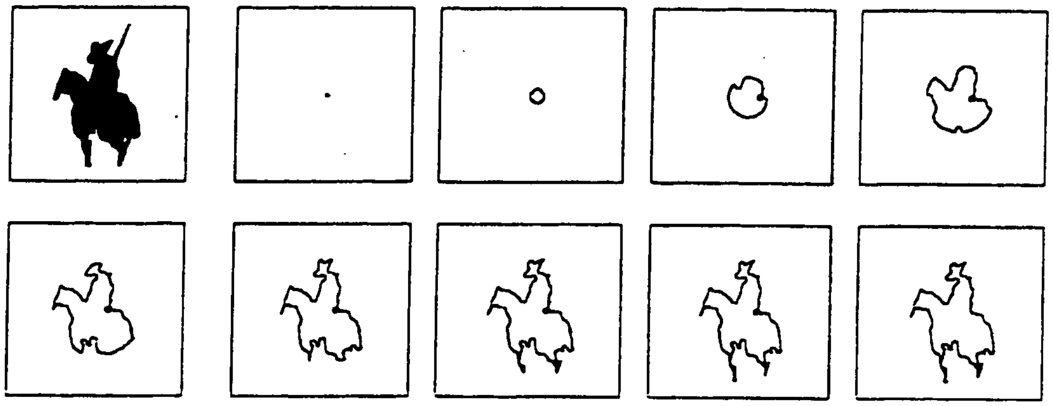


Figure 11.9: This Picasso drawing of Don Quixote is used to illustrate how a level set initialized inside an object with high protrusions can fail to capture its narrow regions, *e.g.*, the lance of Don Quixote.

which is representation of the objects sharing a common boundary, or two objects with close boundaries. In this case the initial level set captures the combination of the two objects, but in the second stage cannot distinguish between two. Figure 11.8b illustrates this point for asymmetrically placed objects embedded a larger object. While the first stage is successful in capturing the external boundary, the second stage fails to capture internal objects due to nearly symmetric initialization constraint. This problem can be solved by placing one active contour per object.

3. Narrow Regions: While an initial motivation of the level set approach was to solve the protrusion problem of the balloon technique, the level sets sometimes fail to penetrate into narrow regions, Figure 11.9. Since the image is initially convolved with a Gaussian (in order to compute derivatives), smoothing by Gaussian (even with a very small one) will simply diffuse away the narrow regions, causing the active contour to stop at their entrance. A possible solution is to initialize a contour outside the object by evolving inwards, requiring further user interaction.

4. Direction of flow of the level set: The user has to initially choose a direction of flow for the level set, either outwards or inwards. We now argue that any particular choice will not be sufficient to capture subtleties in object structure, as illustrated in Figure 11.10, which contains objects with both protrusions and indentations. In order to capture the protrusions, we must initialize the level set outside the objects and evolve it inwards. However, while the level set captures the object with protrusions, it fails to capture the indentations of the other object. A similar argument holds if the level set is initialized inside the objects

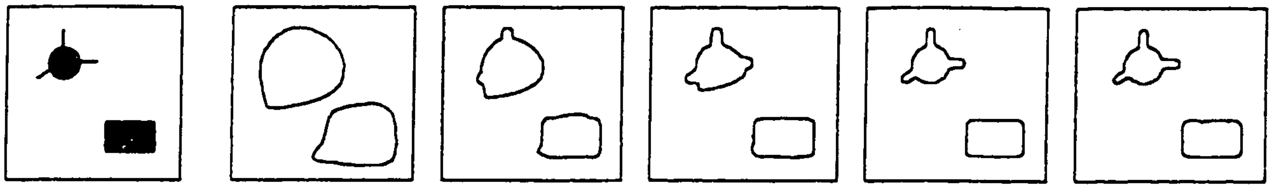


Figure 11.10: This figure illustrates that a particular choice of direction (inward or outward) for the deformation of level sets is not sufficient to capture the detailed structure of objects in an image. In order to capture the protrusions on the top object we must initialize the level set outside the objects to deform in the *inward* direction, as argued in Figure 11.9. However, this level set fails to capture the indentations on the lower object. If we had initialized the level set inside the objects then we would not have captured the protrusions on the top object.

and evolved outwards. Figure 11.9.

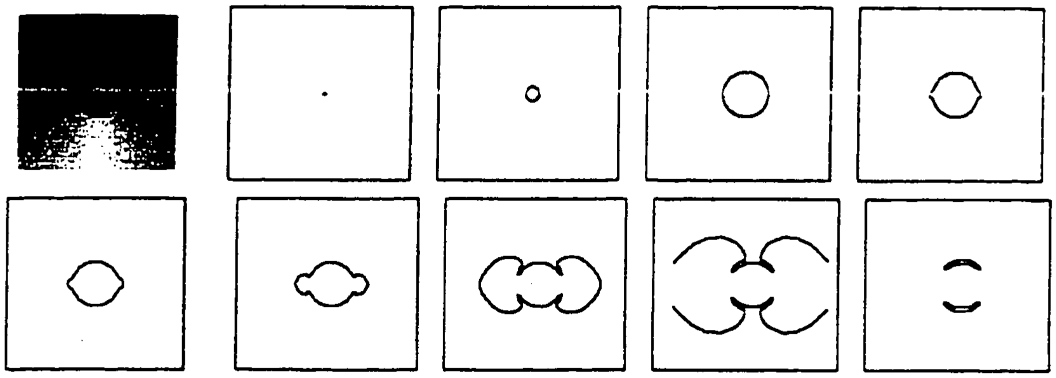
5. Gaps on the boundaries: If there are gaps on the object boundaries. the evolving contour will simply pass over them so that objects represented by incomplete contours are not captured. Figure 11.11a illustrates the gap problem on a synthetic image. The level set does not capture the visible disk with partial boundary segments. This is a serious problem in that in realistic images there are often gaps of information. *e.g.*. in edges of the ultrasound image of Figure 11.11.

6. Automatic initialization: In this approach, a user is required to place initial level sets in the image. Thus, like snakes and balloons. this approach is also semi-automatic in that extensive interaction is required by a knowledgable user if multiple objects are present in the image. It is highly desirable to remove such interaction and dependencies as much as possible. We now present an approach that addresses some of these difficulties.

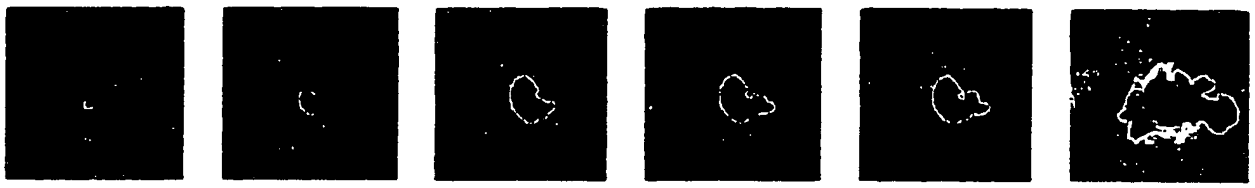
11.2 Reaction-Diffusion Bubbles

In this section, we present a new approach for capturing objects in an image which derives from a shock-based morphogenetic dynamic representation of shape. This representation is formed from a reaction-diffusion process for deforming shape, in the course of which four types of shocks form. This results in a view of shape as a morphogenetic sequence [1] beginning with the birth of fourth-order shocks which grow, and in the process are modified by first-, second-, and third-order shocks, to finally reconstruct the original shape. This view

[1]Much of the inspiration for this view can be found in [115].



(a)



(b)

Figure 11.11: If there are gaps on the object boundaries, the level set will pass through them. (a) the image consists of a disk with a homogeneous interior of intensity 50, while the exterior is radially uniform but linearly increasing in intensity, from 0 on top, to 100 on the bottom, and symmetric on either side. Note that a local region centered on the crossing of the central horizontal line and the boundary of the disk, is a homogeneous region and therefore lacking edge information. As such, the level set passes through these gaps, thereby failing to capture the object. (b) An ultrasound image illustrates the same gap problem in that the level set will cross over the gaps or low intensity gradient regions.

of shape as a morphogenetic process [115, 114, 105, 106] is key to the present approach, and therefore we will review it first. Our current approach is presented as follows: First, we will propose that growing fourth-order shocks, or *bubbles*, represent *hypotheses* for multiple objects in the image. Second, we show that various low-level visual processes can influence the evolution of these bubbles. Therefore, bubbles provide a means for the *integration* of low-level information. Third, we show that the *reaction-diffusion space* of bubbles resolves a number of difficulties as alluded to in Section 11.1.

11.2.1 The Shape From Deformation Framework

The shape from deformation framework [105, 106, 109, 112, 110, 5] proposes to understand shape in the context of a topology of it built around deformations,

$$\frac{\partial C}{\partial t} = \beta(\cdot)\vec{N}, \tag{11.10}$$

where $C(s,t) = (x(s,t), y(s,t))$ is the coordinate vector of points on the curve, s is the length parameter (not necessarily the arclength), t is time, $\beta(.)$ is the deformation function, and $\vec{N}$ is the unit inward normal. Specifically, the following intrinsic, local deformation,

$$\frac{\partial C}{\partial t} = (\beta_0 - \beta_1\kappa)\vec{N}, \tag{11.11}$$

was considered leading to the **reaction-diffusion space** of shapes. Figure 11.12 shows the reaction-diffusion space for the DOLL image. Intuitively, reaction views a shape as a rigid object and breaks it into its components, while diffusion views shape as wax-like material and "melts" the protrusions and indentations, finally taking the shape to a circular point [77]. The reaction-diffusion space is the result of deforming shape by various combinations of reaction and diffusion. **Shocks**, or singularities which form in this space and their signature across the reaction-diffusion space are key to representing shape and are discussed next. It should be noted that they give rise to the **shape triangle** which is a higher-level view of shape as a collection of parts, protrusions, and bends [111].

The set of shocks that form in the reaction-diffusion space together with the time of their formation provides a complete description of shape. These shocks along the reaction axis are equivalent to skeletons [24], but in addition have an associated significance [135], and are classified into four types: A **first-order** shock forms when curvature flows into a curvature extremum and eventually leads to an orientation discontinuity; a **second-order** shock forms when two distinct non-boundary points join, leading to topological changes; a

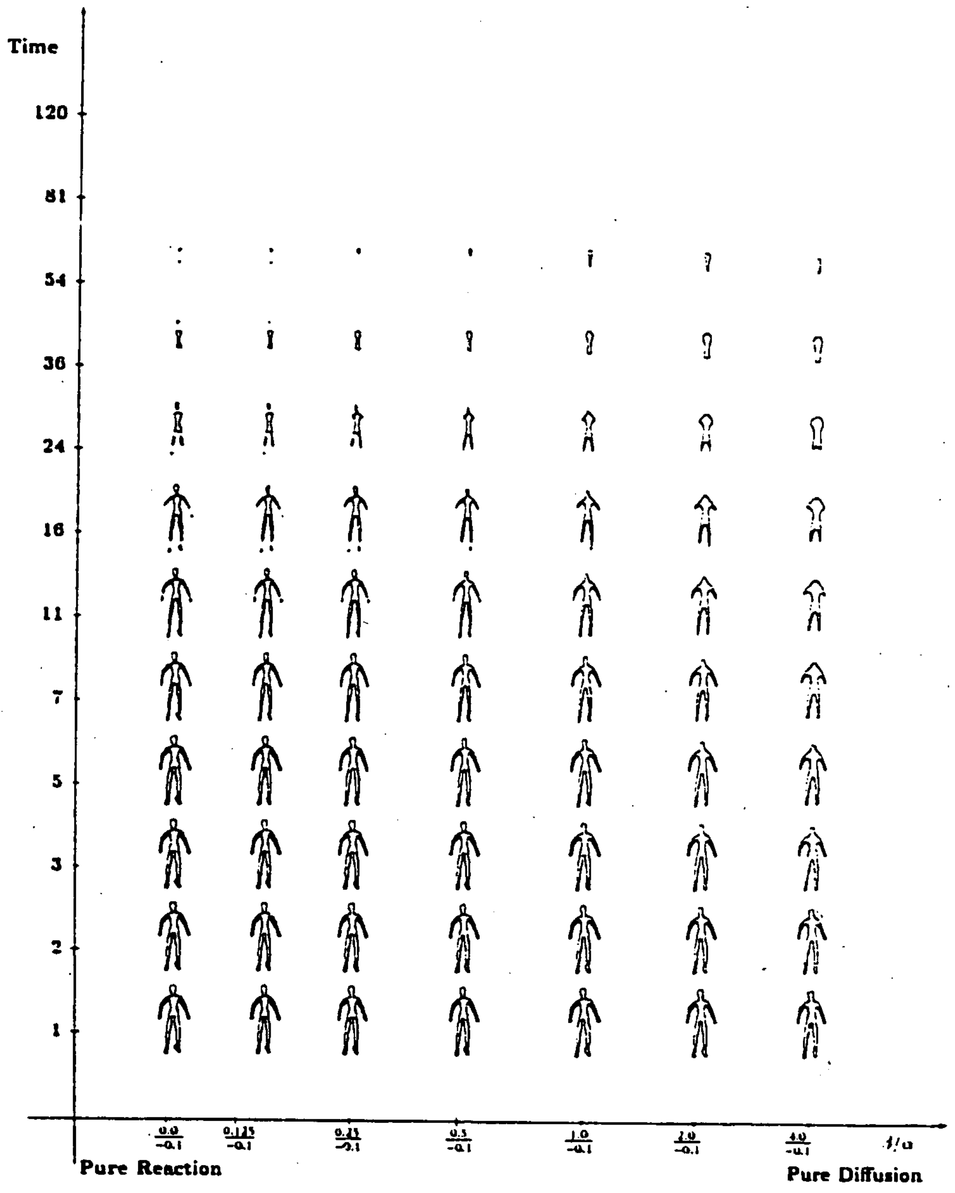


Figure 11.12: The reaction-diffusion space for the DOLL image. The set of shocks that form in the reaction-diffusion space together with the time of their formation provides a complete description of shape.

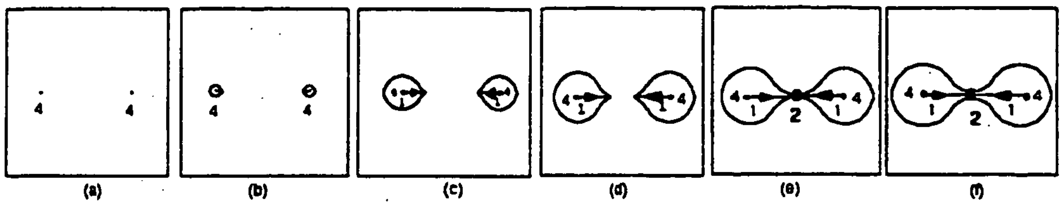


Figure 11.13: This figure illustrates a morphogenetic view of shape as being reconstructed from its shock-based representation. First two forth-order shocks are born, (a), then these shocks grow and are modified by first-order shocks, (b), (c), and (d). These two parts join at the second order shocks, (e) and the shape finally reconstructed, (f).

third-order shock forms when two separate pieces of the boundary meet along a contour: a **fourth-order** shock forms when an entire closed boundary collapses into a single point.

The numerical implementation of this process faces a number of fundamental difficulties, *e.g.*, topological splitting, built-up of error, capturing discontinuities, *etc.* Osher and Sethian, in application to flame propagation, proposed that curve evolution can be considered as the evolution of a surface, $\phi(x, y)$, [154, 182, 181]. The reaction-diffusion space can then be generated by simulating

$$\phi_t + (\beta_0 - \beta_1\kappa)|\nabla\phi| = 0, \qquad (11.12)$$

where κ is the curvature of the level set $\phi(x, y) = 0$. In addition, to capture and maintain discontinuities, "shock-capturing" numerical schemes are required [80, 120, 152, 154, 183, 193]. For a review of this class of numerical schemes applied to computer vision problem see [206].

We should also take note of a rigorous and elegant approach by Alvarez, Guichard, Lions, and Morel who based on certain axioms for image processing motivate similar PDE evolutions and study the existence and uniqueness of their viscosity solution [5, 6, 7].

11.2.2 Bubbles As Multiple Object Hypotheses

Our approach to capturing objects in an image is based on the shock-based view of shape described above. Specifically, a known shape gives rise to a shock-based representation which can then be used in *reverse time*, as a morphogenetic sequence to reconstruct it, Figure 11.13. Typically, however, the segmentation of shapes in an image is a fundamentally difficult problem, so that complete information about figures as distinct from their back-

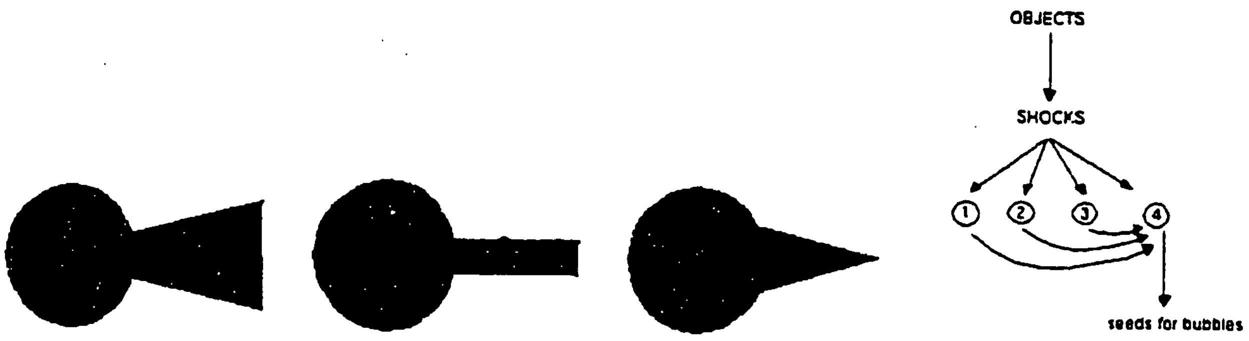


Figure 11.14: Under the reaction-diffusion process, every object generically leads to at least one fourth order shock, as illustrated for three shapes.

ground is typically not available. The question, therefore, is how to obtain a shock-based representation for these shapes *directly* using only the partial information that differentiates figures from their background. Observe that fourth-order shocks represent components of an object, in that their birth initializes evolving contours, which will be modified by other types of shocks. As such, placing a fourth-order shock in an image is tantamount to *hypothesizing* an object component. Since fourth-order shocks are generally situated centrally in these components, and are distant from their boundaries, we initiate them in the homogeneous areas, away from differential structure.

Ideally, first-, second-, third-, and fourth-order shocks ought to be initialized to form object hypotheses. However, note that first-order shocks are not isolated: they give rise to other types of shock, and eventually terminate in fourth-order shocks. Furthermore, second-order shocks do appear in isolation, but they eventually lead to fourth-order shocks. Third-order shocks, on the other hand, are the limits of fast moving first-order shocks, and as such the above comments regarding first-order shocks also hold for them. Therefore, every object generically[2] leads to at least one fourth-order shock, Figure 11.14. Thus, we initialize only fourth-order shocks for image segmentation.

Image-based information from low-level visual processes is then used to validate, annihilate, or modify these initial hypotheses. The fourth-order shocks in their initial stage of growth resemble "bubbles" which grow, shrink, merge, split, and disappear. When a portion of the boundary of a bubble approaches differential structures, as indicated by measures such as high gradients of intensity (edges), texture, stereo disparity, optical flow, *etc.* , it

[2] "Generic" refers to a representation that is stable under slight def rmations.

slows down, while other portions of its boundary deform without inhibition. When two growing bubbles collide, they merge; this indicates that two separate object hypotheses have merged to represent one object hypothesis. The opposite phenomenon occurs when a portion of a bubble collides with another portion, splitting it into two bubbles. This indicates that a single object hypothesis, driven by image forces, has given birth to two object hypotheses. It is also possible for several bubbles to merge to create a "negative bubble" that shrinks, either to objects embedded inside, or otherwise to annihilation. We now turn our attention to the way low-level visual processes can be integrated by affecting the deformation of bubbles.

11.2.3 Integration of low-level visual processes

The above phenomenon for the deformation of bubbles is governed by a reaction-diffusion process used in representing shape [112], $\phi_t + (\beta_0 - \beta_1\kappa)|\nabla\phi| = 0$. That such a process for a *single* deformable curve can be used with *proper initialization* to capture object structure in images containing *complete boundaries* with associated high intensity gradients was shown by Caselles [33] and Malladi [133, 134], equations (11.7) and (11.8). However, in realistic images, intensity edges are often incomplete in describing the boundary of objects. In fact, this incompleteness is not exclusive to the process of edge detection: other low-level visual processes, such as texture, stereo, optical flow, *etc.* , often capture certain differential structures, while missing others completely. The intensity gradient map and the stereo disparity map of the MANNEQUIN image, Figure 11.15, illustrate this point: while the visual process of edge-detection captures the boundary between the posterior mannequin and the background, it misses the boundary between the two mannequins which is captured by the stereo disparity map process. In this case, the two gradient maps, when taken together, serve to capture boundaries in the original image. Similarly, the intensity gradient map and the optical flow magnitude gradient also depict this point: while the edge detection process captures some of the background structure, the occluding contour of the fish is only present in the optical flow results. To summarize, low-level visual processes taken in isolation provide *partial* information about the differential structure in the image. Bubbles can integrate this information by using the low-level maps to guide their deformation. Specifically, the flow of bubbles by the reaction-diffusion process can be modulated by the presence of such information: bubbles deform unless inhibited by at *least one* of these low-level processes:

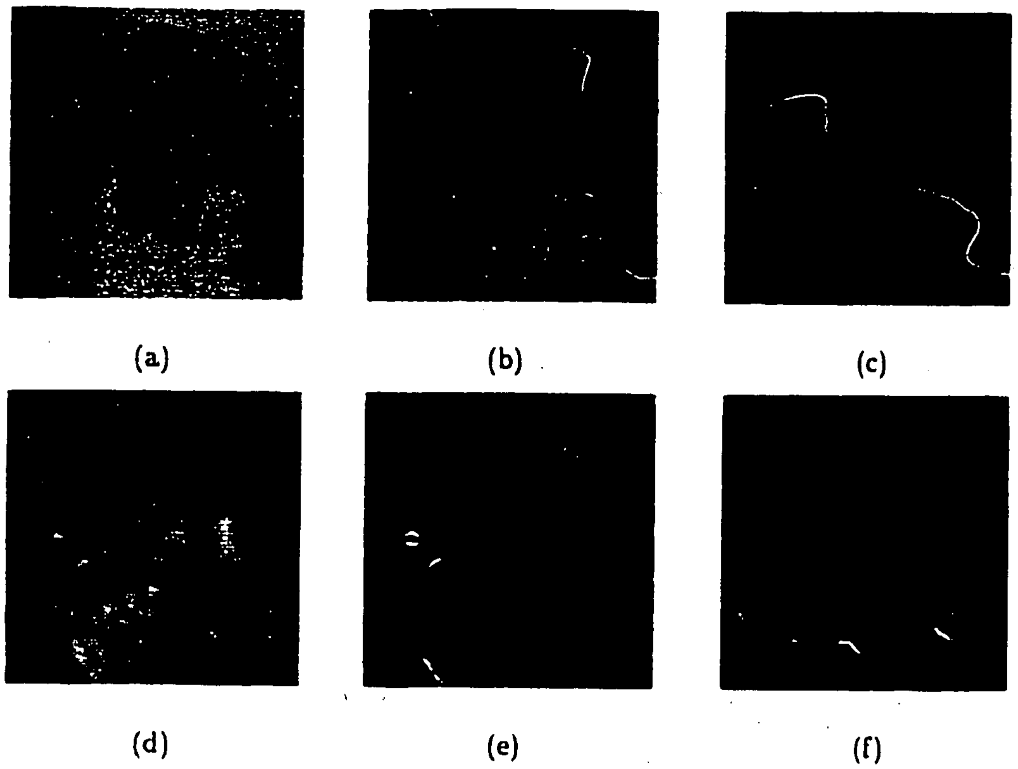


Figure 11.15: This figure illustrates how low-level visual processes provide *partial* information about differential structure in the image. (a) original MANNEQUIN image, (b) intensity gradient map, (c) stereo disparity gradient map [163]. Note that each gradient map captures only some of the occluding contours. (d) original WANDA image, (e) intensity gradient map, (f) optical flow gradient map [162]. Again, the two gradient maps (e) and (f) again provide only partial, but complementary information. Bubbles can integrate the results of such processes.

$$S(x,y) = 1 - max(S_1, S_2, ..., S_n), \tag{11.13}$$

where $S_i(x,y)$ represents a function that indicates differential structure, where zero indicates a homogeneous region and one indicates a high gradient, and i represents various low level visual processes, such as edge detection, optical flow, stereo disparity, texture, *etc.* , leading to the flow equation

$$\phi_t + S(x,y)(\beta_0 - \beta_1\kappa(x,y))|\nabla\phi| = 0. \tag{11.14}$$

While the integration of early visual channels fills many gaps of information, some gaps remain since no evidence of differential structures is present in any channel. These gaps are dealt with by the r action-diffusion space and the shock-based representation, as described next.

11.2.4 C nvergence and Interaction of Bubbles

As examples show, objects boundaries are captured by pairs of bubbles. For example, in Figure 11.22, the sphere is captured by a bubble from inside, and another on the outside which captures the sphere and its shadow. Two objections may be raised at this point. First, it appears that objects are represented by multiple results. Second, the inner and outer bubbles appear to be a few pixel apart, resulting in in-accuracy problems. We believe that multiple representations are in fact necessary when objects interact closely with each other. For example, the sphere and its shadow result in three bubbles: one for the sphere, one for the shadow, and one for both as a group. The converged bubbles, thus, do not represent boundaries themselves but represent objects. The boundaries themselves can be obtained by using each converged bubble as an active contour whose convergence results in the final boundary.

Second, an important issue facing a curve evolution approach is convergence. It is often the case that bubbles flow over weak boundaries if the evolution is allowed to continue for a very long time. We have followed two approaches to this problem. First, when two bubbles (or portions of) are very close in Hausdorff distance and do not change for many iterations they are considered *locally converged*. Since it is often the case that one converged bubble crosses the underlying boundary while waiting for the convergence of another bubble, the local convergence criterion resolves most of the cases. It is also possible to incorporate a snake-like measure as [98, 35] have proposed. A second approach that represents ongoing research is to relate convergence to a measure of when the inter-bubble shocks coincide with localized gradients. Returning to the question of accuracy when double bubbles converges to the objects 1-2 pixels apart, we typically deform each separately for a few iteration (*e.g.* using snakes type models as in appendix A). In addition, Siddiqi and Kimia [197] have recently developed subpixel methods which allows convergence up to and include the very same pixel while keeping bubble apart using the shock approach described above.

11.2.5 Reaction-Diffusion Space for Bubbles

The reaction-diffusion process for representing shape involves two distinct extremes: on the one hand, intuitively, the reaction pr cess views shape as a rigid mass whose components are then broken off; on the other hand, diffusion "melts" the boundary of the shape by propagating and amalgamating boundary information, finally converging it onto a circular point. Similarly, the deformation of bubbles in the reaction-diffusion space involves two extremes: on the one hand, in the reaction process, the bubbles are breakable structures that can easily form singularities, grow into narrow straits, and shrinks onto sharp pointed structures. On the other hand, in the diffusion process, bubbles are cohesive structures that do not easily form singularities, and as such do not easily go through gaps, nor respond to small ripples or noise on the object boundaries. The full space, therefore, represents a spectrum of structures that is captured from the image: the reaction process captures *detailed* structure from *complete* information provided by low-level processes, while the diffusion process captures *coarse* boundary structure from *incomplete* information, *e.g.*, boundaries with small gaps, Figure 11.17. This is also clearly illustrated for the Kanizsa images, Figure 11.20, and an ultrasound image, Figure 11.21. In addition to *dealing with gaps* and providing a *coarse-to-fine representation* for the boundary of captured objects, the interaction between reaction and diffusion also places the captured boundaries in a *hierarchy of significance*, since small boundaries disappear faster than larger size boundaries along the diffusion axis, Figure 11.17.

11.3 Experimental Results

In this section, we discuss implementation issues, present a number of experiments, and illustrate the effectiveness of this approach on several examples. The implementation of the reaction-diffusion bubble segmentation technique involves four steps: smoothing the image, initializing bubbles, simulating their deformation, and stopping their evolution upon convergence.

First, the image is smoothed by a shock-based nonlinear diffusion technique [103], to remove small-scale features, principally to speed up the convergence of bubbles. In contrast to Gaussian smoothing, this technique does not blur across sharp edges.

Second, a large number of bubbles are randomly initialized in the homogeneous areas of the image. Specifically, a uniformly random map of bubbles is first generated, and then modified by removing bubbles which have a high standard deviation from their intensity mean, thus, indicating a non-homogeneous region. While this has proven sufficient for our purpose, we intend to improve the results by initializing bubbles using a probability function that is dependent on a more sophisticated measure of homogeneity.

Third, the technique is simulated by solving equation (11.14) with an initial surface $\phi(x, y, 0)$, constructed from the distance transform of the initial bubble contour. The numerical simulation involve an upwind Essentially Non Oscillatory (ENO) numerical scheme [80, 152, 120, 154, 183, 193]. For a review of shock capturing schemes for computer vision problems, see [206]. Note that initially diffusion shrinks very small bubbles to annihilation. As such, diffusion is turned on only after some initial growth has taken place, Figure 11.24.

Fourth, the evolution process is stopped when changes in two subsequent iterations are negligible. Specifically, we currently use the 2-norm difference between two adjacent iterations of the bubble image, $||\phi(., t + \Delta t) - \phi(., t)||_2$ to measure changes. While this criterion has performed well in most cases, the change measurement can be improved if the contours are first captured and then their difference is used.

Finally, contours in the image are traced from results in the reaction-diffusion space and placed in a hierarchy of significance.

A question naturally arises as to what effect the randomness of initialization has on the final result. Since the bubbles converge onto image structure we expect that given a large number of bubbles, the final outcome will not be sensitive to how they are initialized. Our experimental results confirm this: Figure 11.17 shows the reaction-diffusion space for an MRI image; Figure 11.18 repeats the experiment for the reaction axis with six different initializations; it is evident that they all converge to the same structure.

We now illustrate the effectiveness of our method on a variety of images, shown in Figure 11.16. Figures 11.17, 11.19, 11.20, 11.21, and 11.23 show the reaction-diffusion spaces for these images. We discuss below how the reaction-diffusion bubbles resolve some of the problems associated with previous approaches:

Symmetric initialization: The initialization of a *large* number of bubbles in the *homogeneous* areas of the image solves the symmetric initialization problem. For example, Figure 11.17 shows that bubbles are placed nearly symmetric with respect to object boundaries, resolving difficulties depicted in Figure 11.7. Figure 11.18 confirms that this is indeed the typical case.

Multiple initialization: It is evident that a large number of randomly initialized bubbles also resolve the multiple initialization problem, capturing embedded structures. For example, Figure 11.22 illustrates that bubbles capture the sphere and its attached shadow, resolving difficulties presented in Figure 11.8. Figure 11.17 also shows that bubbles capture several structures, such as the skull as well as the embedded ventricle.

Narrow region and direction f evolution: These problems are solved by the use of bubbles simultaneously growing outside and inside the object of interests. Observe how in Figure 11.19, the lance of Don Quixote is captured by the bubbles initialized outside the shape while the narrow indentation in the body of the horse is captured by bubbles initialized inside the shape.

Gaps on the boundaries: This problem is dealt with by the diffusion process, which captures the coarse level structures, effectively integrating across gaps. This is illustrated in the Kanizsa image, Figure 11.20, and in an ultrasound image, Figure 11.21.

Automatic Initialization: The bubble technique is an automatic process in that no user interaction is required. Since bubbles are placed randomly, the evolution requires no user input on the initialization. Also, the parameters used in our method are fixed and uniform for a variety of images: 1) *standard-deviation* is a threshold of homogeneity, 2) *time-threshold* is used stop the evolution of bubbles if changes are small, 3) *gradient-power m*, defined in $\frac{1}{1+|\nabla G_\sigma * I(x,y)|^m}$, is typically chosen between 1.5 and 2.0. This defines all parameters related to the deformation of bubbles.

While these preliminary results show the effectiveness of our approach, a number of problems remain to be resolved: First, bubbles often collide on the boundaries whose edge strength is very weak. Second, in very noise images, bubbles move slowly and sometimes fail to capture object boundaries. A multiscale approach may resolve this problem and we are currently investigating this approach. The current results, however, are promising and we expect that the reaction-diffusion bubble technique will be useful in the medical, industrial, and other applications.

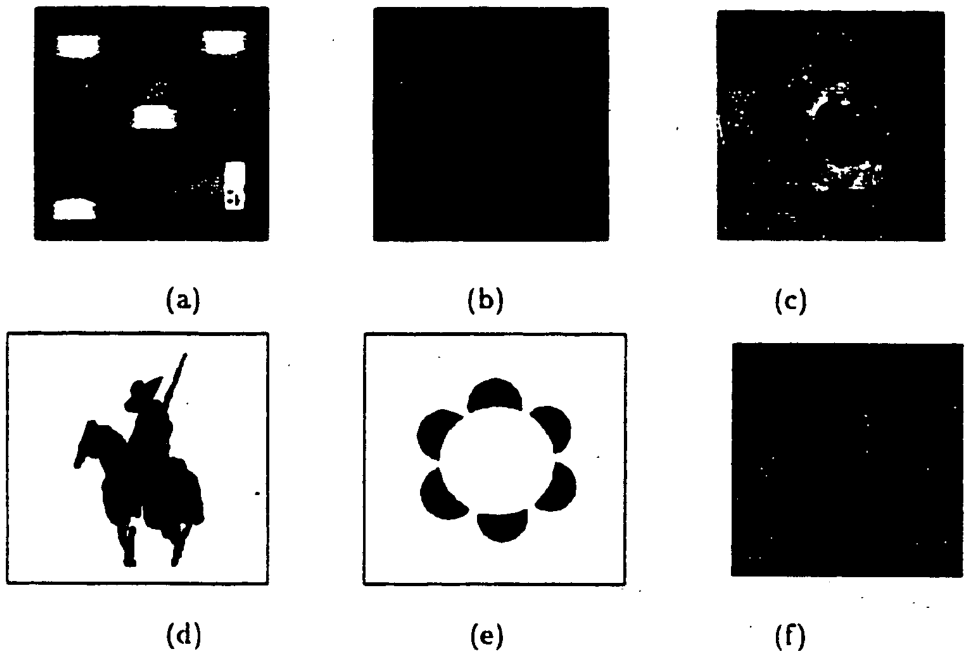


Figure 11.16: The original images used in our experiments: (a) a range image from The National Research Council of Canada's Laser Range Image Library; (b) an intravascular ultrasound image; (c) a MRI image of brain; (d) a binary image from the Picasso drawing, don Quixote. (e) a synthetic Kanizsa image. (f) a gray scale image.

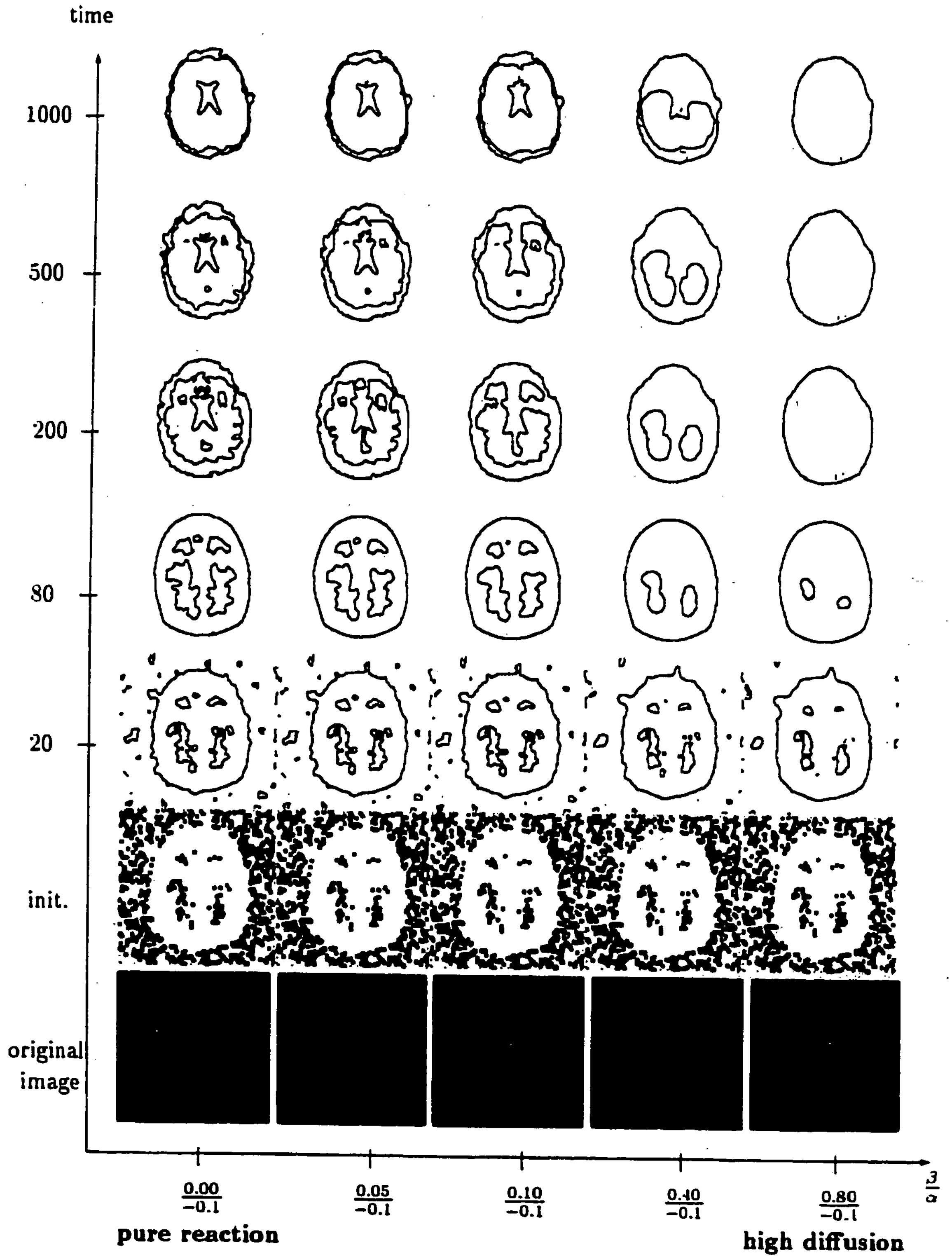


Figure 11.17: This figure shows the reaction-diffusi n process for bubbles on an MRI brain image. Each column depicts an evolution from the original imag on th bottom row, using a random initialization, converging to the image, on the top row. The left column is evolution by reaction, the right by high diffusi n, and the intermediate columns, by combinations.

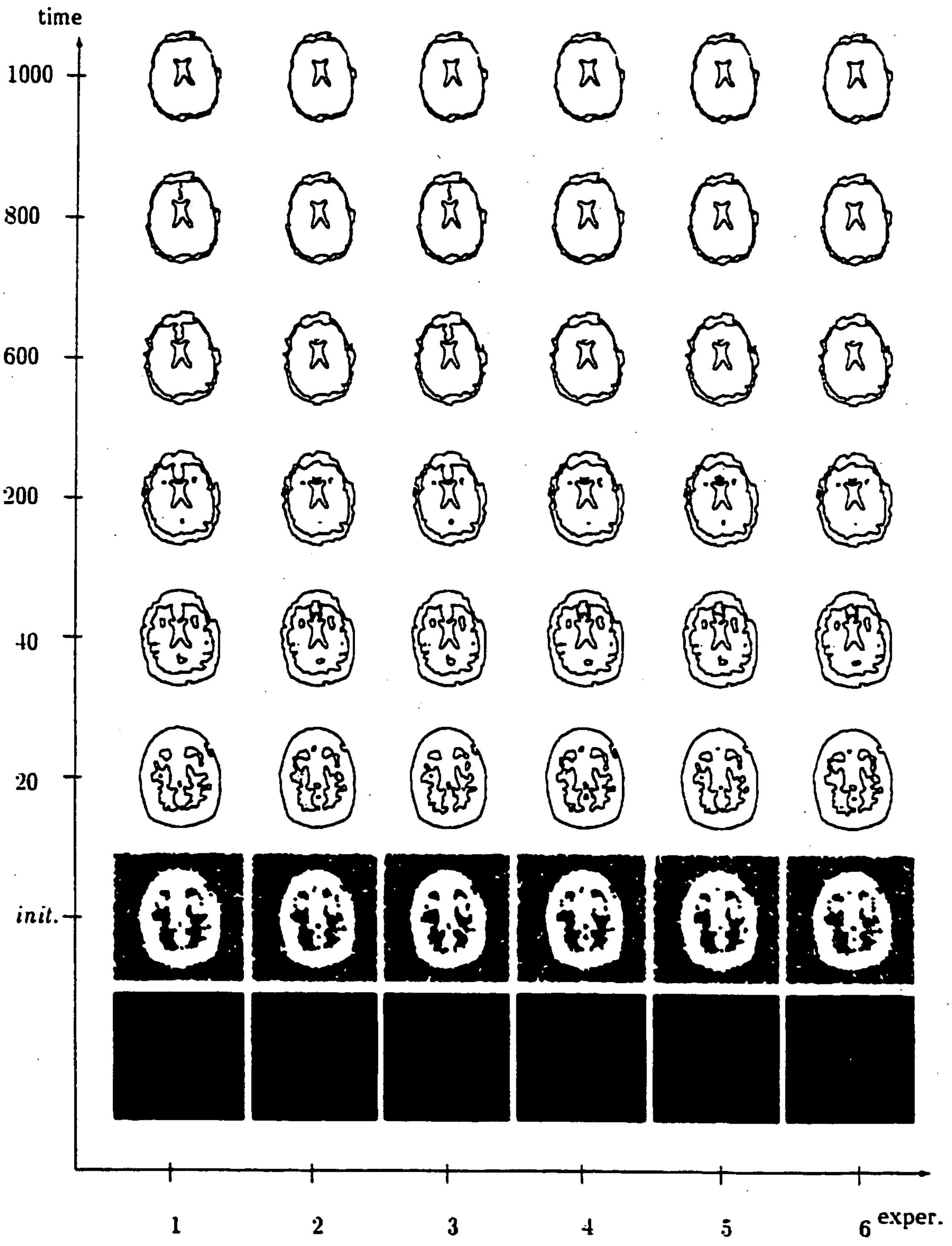


Figure 11.18: This figure illustrates that the random initialization does n t affect the final results of this process. Note how the reaction process integrates the random bubbles to converge onto the structures determined by the image.

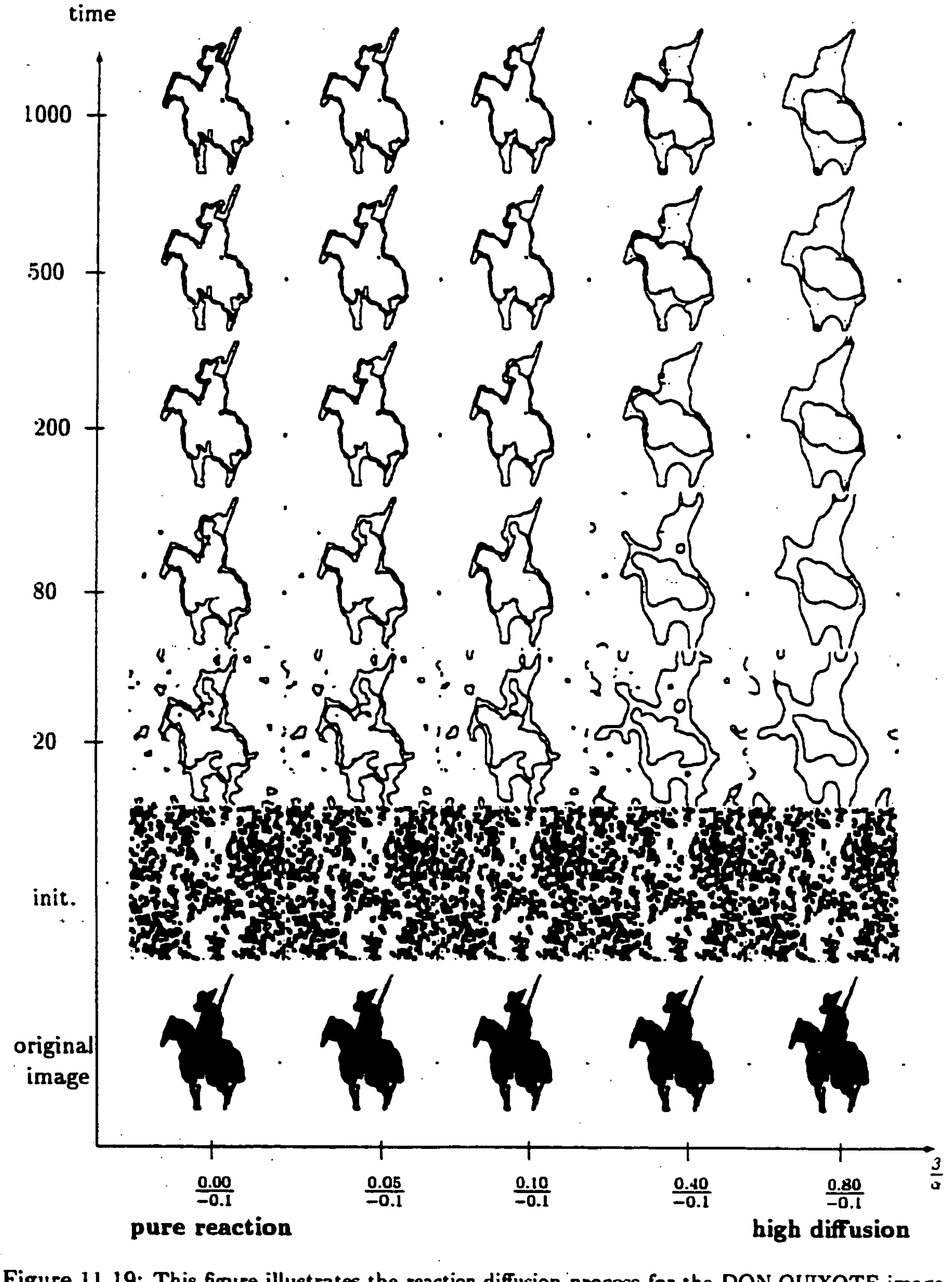


Figure 11.19: This figure illustrates the reaction-diffusion process for the DON QUIXOTE image. Each column depicts an evolution from the original image on the bottom row, using a random initialization, converging t th image, n the top row. The left column is evolution by reaction, the right by high diffusion, and the intermediate columns, by combinations.

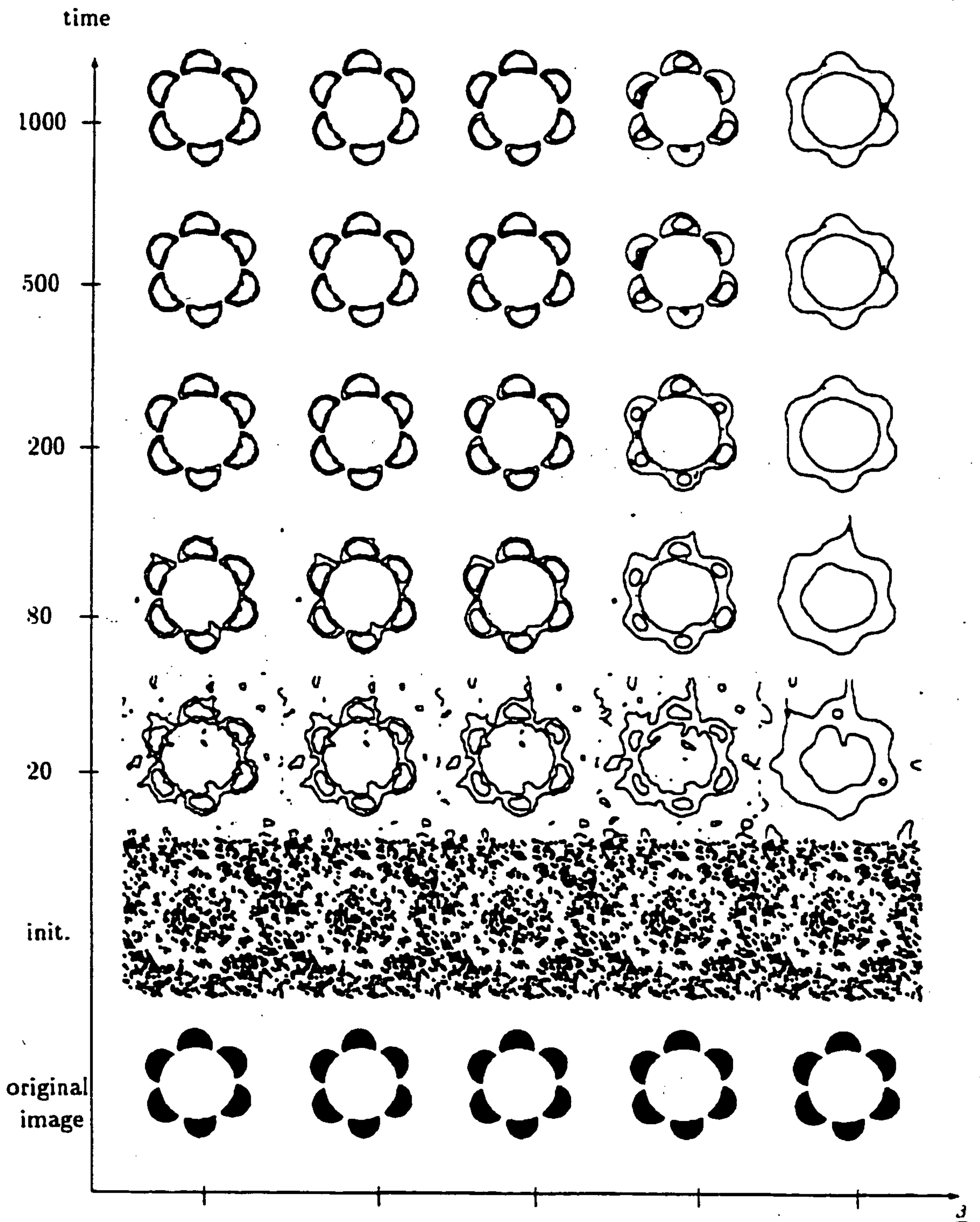


Figure 11.20: This figure illustrates the reaction-diffusion space for a Kanizsa image. Note that the process of high diffusion does n t allow bubbles to pass over the gaps. Each column depicts an evolution from the original image on the bottom row, using a random initialization, converging to the image, on the top row. The left column is evolution by r action, the right by high diffusion, and the intermediate columns, by combinations.

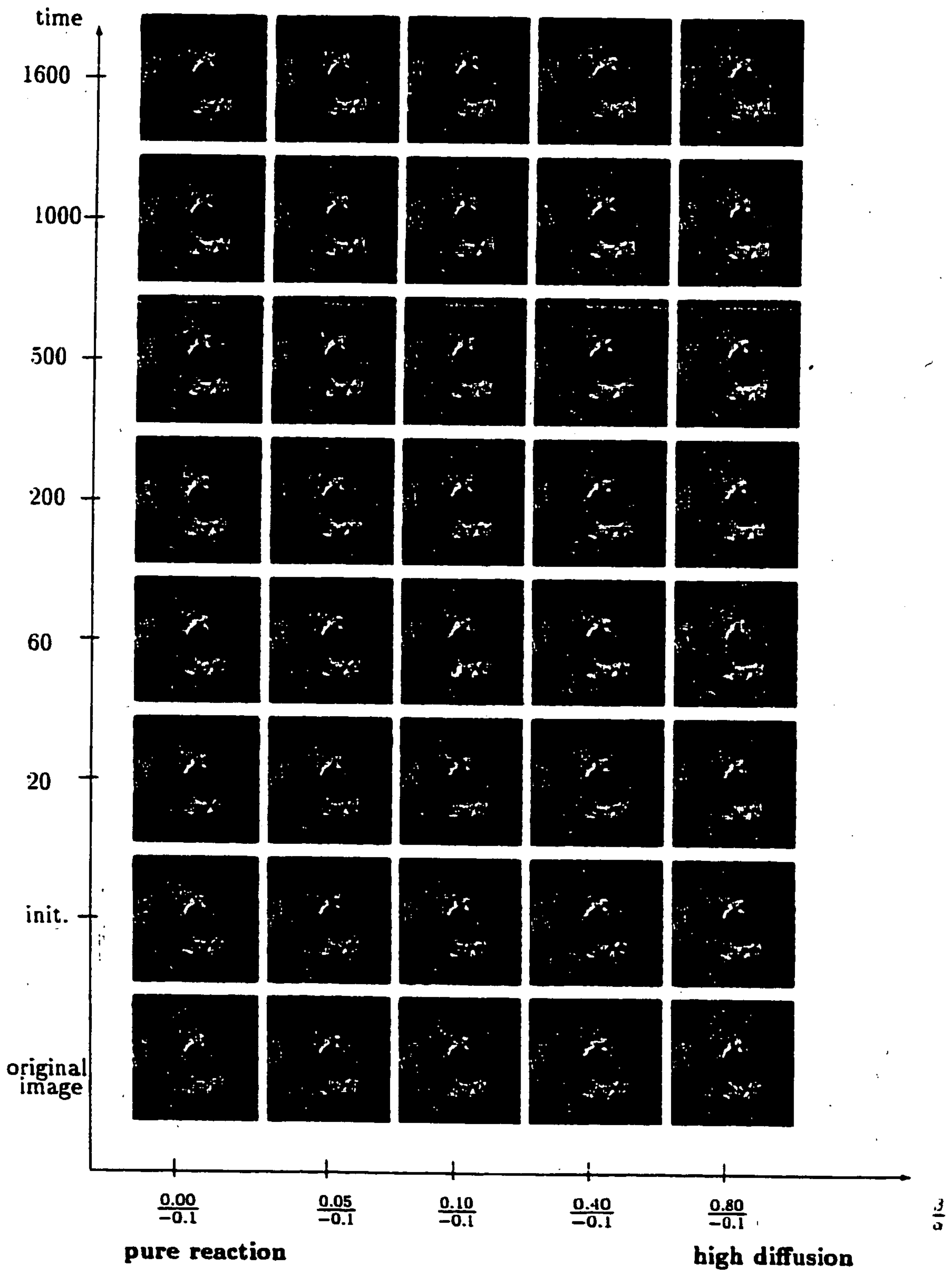


Figure 11.21: The reaction-diffusion process is shown for an ultrasound image in order to illustrate the importance of diffusion. Each column depicts an evolution from the original image on the bottom row, using a random initialization, converging to the image, on the top row. The left c lumn is evolution by reaction, the right by high diffusion, and the intermediate columns, by combinations.

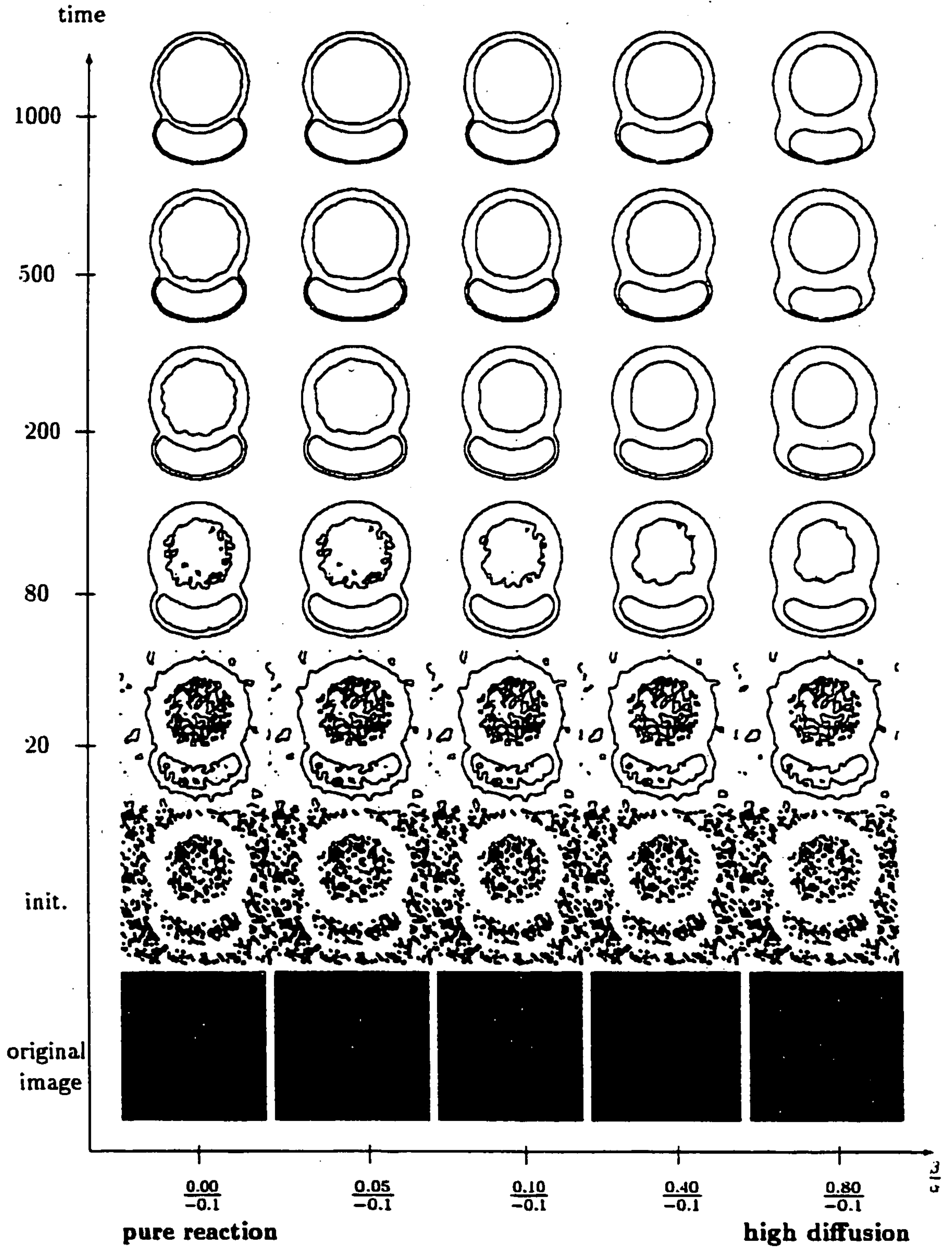


Figure 11.22: This figure illustrates the reaction-diffusion process for the SPHERE image. Each column depicts an evoluti n from the riginal image on the bottom row, using a random initialization, converging to the image, n the top row. The left column is evolution by reaction, the right by high diffusion, and the intermediate columns, by combinations.

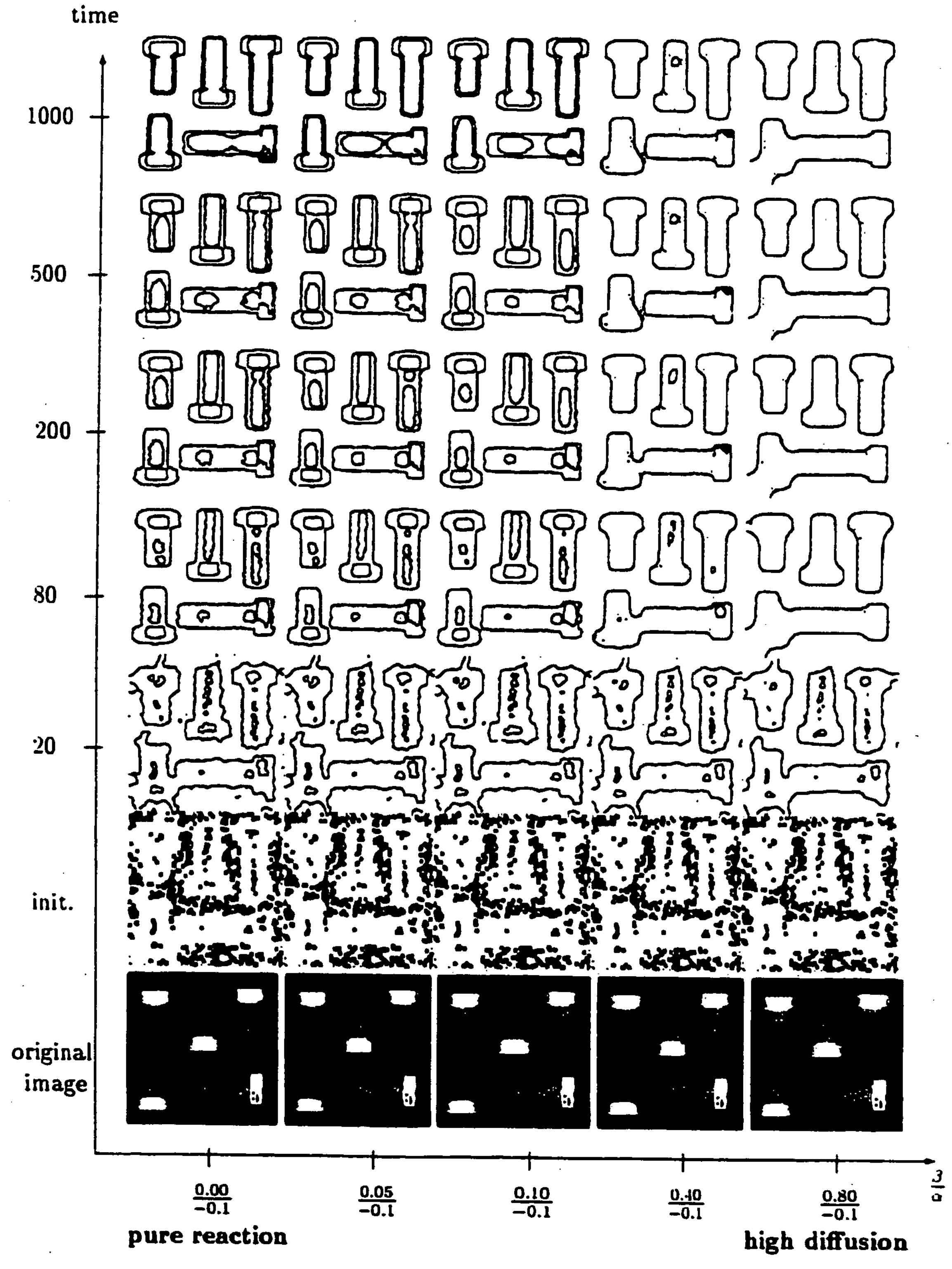


Figure 11.23: This figure illustrates the reaction-diffusion process on a TOOLs image. Each column d picts an evolution fr m the riginal image on the bott m row, using a random initialization, converging to the image, on the top row. The left column is evolution by reaction, the right by high diffusion, and the intermediate columns, by combinations.

Chapter 12

3D Reaction-Diffusion Bubbles

This chapter is organized as follows. In Section 12.1, we present three approaches to the three-dimensional image segmentation problem and generalize the 2D reaction-diffusion bubbles method to 3D. In Section 12.2, we consider a notion of diffusion and construct reaction-diffusion space for 3D bubbles. Finally we illustrate the technique for several synthetic and medical images.

12.1 Three Dimensional Bubbles

The generalization of the 2D reaction-diffusion scheme to segmentation of 3D images faces two major issues. We will address the issue of gaps in the next section. In this section we will consider three approaches to applying bubbles for 3D segmentation, in analogy to [48].

12.1.1 3D Segmentation from 2D Bubbles

The most straightforward approach is an independent segmentation of each slice along some axis. However, we now show that 1) large gaps in boundaries of one slice lead to poor segmentation results; 2) a segmentation of a slice along a different axis may lead to different results: and 3) the reconstruction of the surface and its properties from 2D contours may lead to inaccurate results. Figure 12.1 illustrates that when adequate information is available, the slice-by-slice 2D segmentation is quite good and the results are satisfactory. In realistic imagery, the small gaps in imperfect edge maps are bridged by the diffusion component of the reaction-diffusion space. While in certain 3D images, 2D diffusion can accomplish also this, Figure 12.2, in others where the gaps is extended in the slice plane (as maybe the case in MR images due to the interaction of nearby elements), 2D diffusion is no longer sufficient,

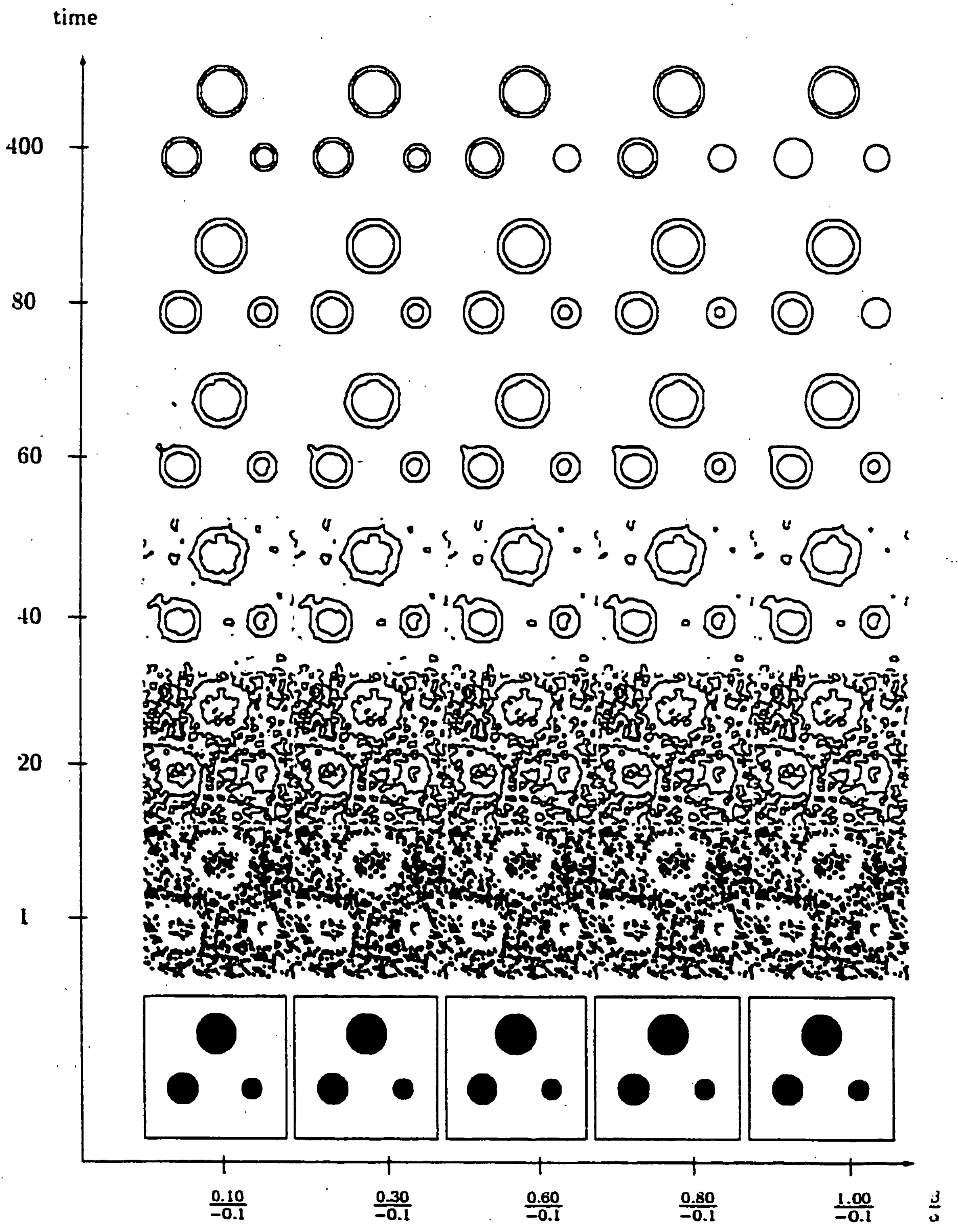


Figure 11.24: This figure illustrates that the size of the object is very important in the diffusion process. Diffusion, β is increased quadratically to prevent the bubbles from being swept away bef re they can grow at all.

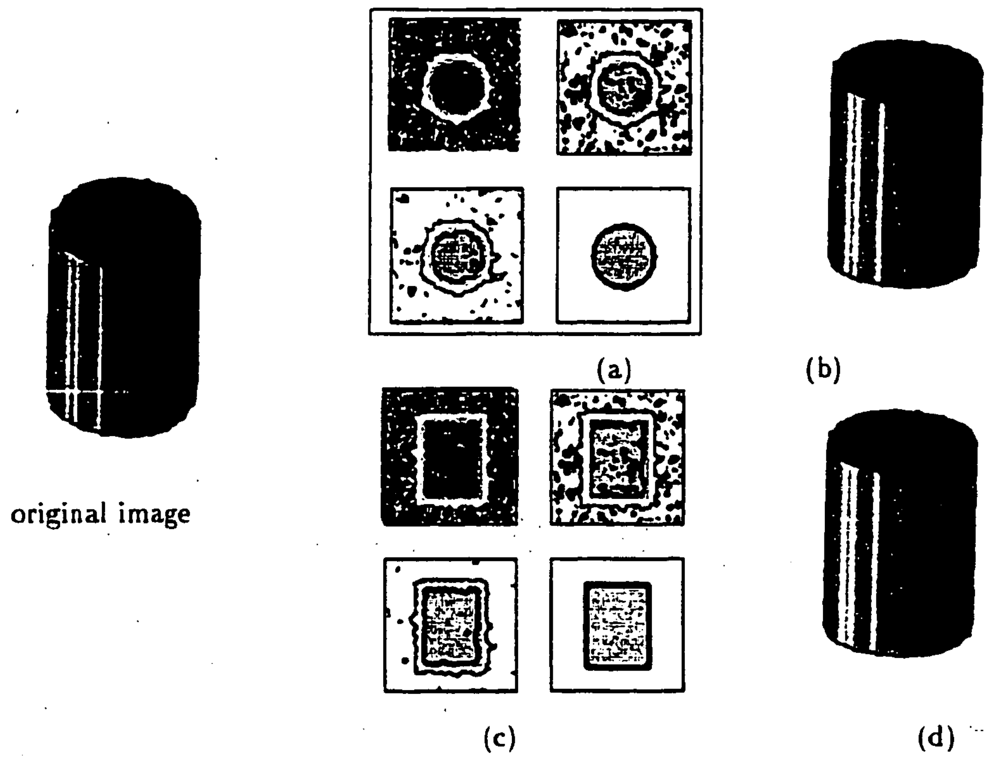


Figure 12.1: This figure illustrates that two-dimensional reaction-diffusion bubbles lead to successful segmentation of a synthetic three-dimensional data set (64x64x64) of a *cylinder* with no boundary gaps. (a) the use of 2d bubbles to segment the cross section along the z-axis; (b) final segmentation; (c) the use of 2D bubbles along the x axis. (d) The final result. In this case, the results are identical.

Figure 12.3. In addition to the gap problem, Figure 12.3 also points to the extrinsic nature of the segmentation: when the image data satisfies the segmentation model assumptions perfectly, the 2D segmentation results are invariant to the slicing direction. However, when the image data violates some of these assumptions, it likely does so differently in each slicing direction. Thus, three 2D segmentations of the same 3D image along x, y, and z directions is likely to produce different results. For example, Figure 12.3 illustrates that the diffusion process only captures gaps which do not lie in the slicing plane, rendering the process extrinsic. A third drawback with this approach is the reconstruction of the surface or its properties (*e.g.*, volume) from segmented contours which is not a trivial problem; when changes between adjacent slices are large compared to the slice thickness, inaccuracies can arise from the reconstruction.

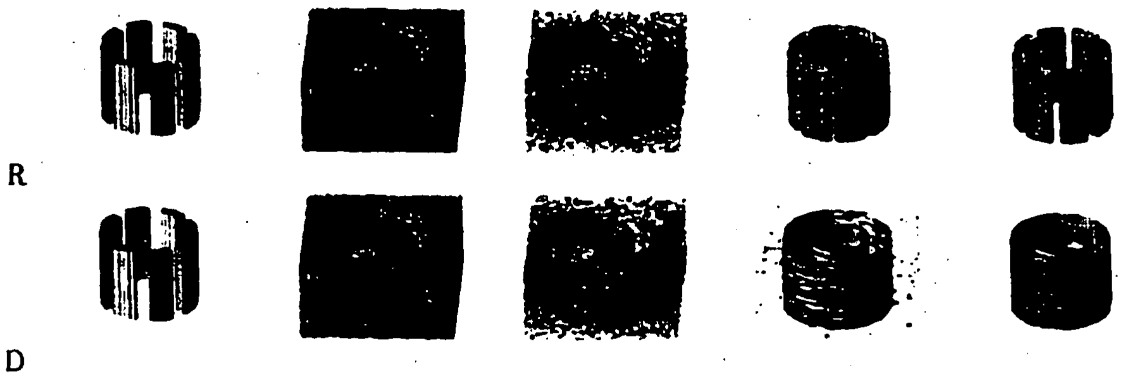


Figure 12.2: This figure illustrates the use of the reaction-diffusion space of 2D bubbles to deal with small gaps in each slice. Note that first column (R) depicts the original image and the second column (D) shows the initialization of bubbles; the evolution is shown from left to right; the top row indicates reaction while the bottom row indicates diffusion. While the reaction process captures individual surface segments, the diffusion process closes gaps and captures the overall shape.

12.1.2 3D Segmentation from $2\frac{1}{2}$D Bubbles

Independent intra-slice segmentation ignores the inter-slice continuity inherent in a 3D data set containing coherent objects. Observe that in Figure 12.3 the missing edge structure contained in one slice is not consistent with information in its neighboring slices. Therefore, it seems appropriate to regularize edge information by using 3D edge operators, *e.g.*, the 3D edge intensity gradient,$|\nabla_3 I|$ or a 3D edge operator. Indeed, Cohen and Cohen [48] use 3D edge data to guide 2D balloons in each slice. However, regularization of edges can deal with gaps which are very limited in one dimension, *i.e.*, depth. Even in such cases, the closing of gaps is at the expense of blurring sharp structures, as well as completely missing small structures, since large 3D edge operators do not respond to them. For example, for the "grooved cylinder" of Figure 12.3, while this method would capture the overall shape, the final surface would be rounded and the small grooves not represented. A successful segmentation should report both small and large scale regions; in this sense, this approach lacks a notion of scale in 3D, Figure 12.4. In addition, the solution is extrinsic and the reconstruction problem is not resolved. The difficulties associated with the extrinsic two-dimensional bubbles and diffusive $2\frac{1}{2}$D bubble methods lead us to use three-dimensional bubbles, as follows.

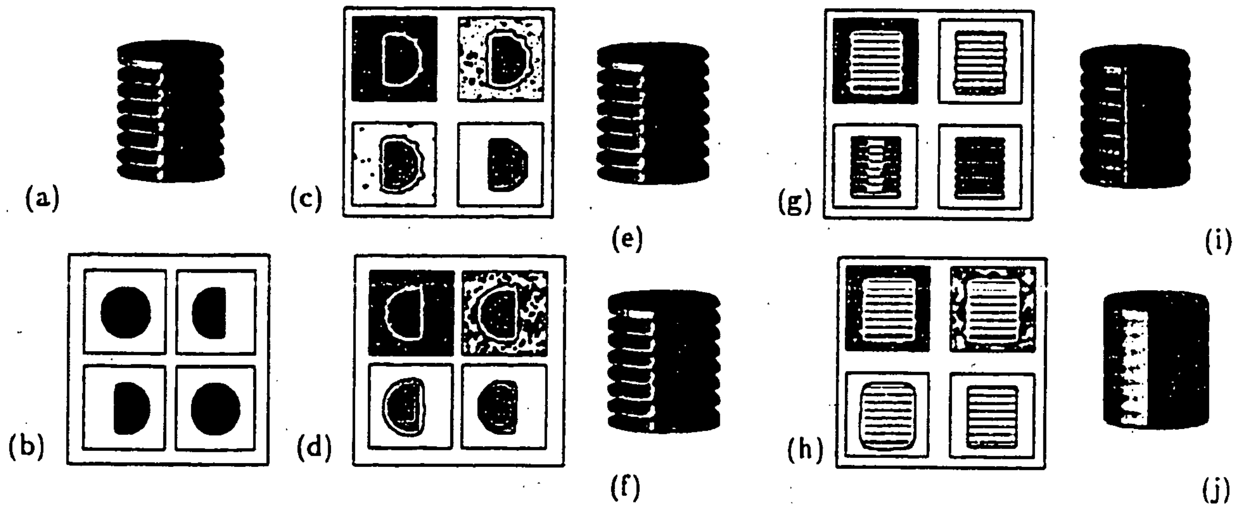


Figure 12.3: 2D bubbles and gaps (II): (a) original image, (64x64x64) simulates a cylinder with many missing edges (grooves); note that slices are grossly distorted(b); (c-d) reaction and diffusion of 2D bubbles cross the missing edges as shown in 3D in (e-f), respectively. However, reaction (g) and diffusion (h) along a different slicing of the 3D image captures the detailed (i) and the overall shape (j), illustrating the extrinsic nature of this process.

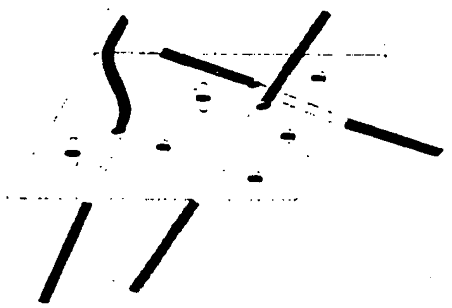


Figure 12.4: This figure is a caricature of a cross-section of a structure resembling vessels and nodules. Note that the insignificant blobs in the 2D plane may be the cross section of significant 3D structure (veins) or may be noise; a notion of scale in 3D is required.

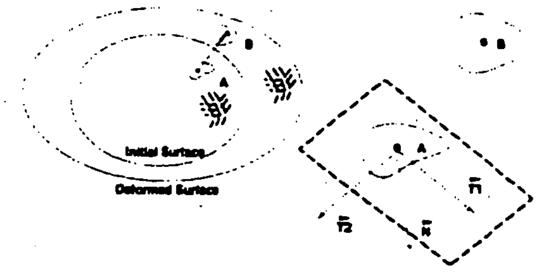


Figure 12.5: Deformation of a surface can be captured in its local coordinate frame. Point A on the initial surface is moved arbitrarily to point B. The displacement $\vec{AB}$ is represented in the $(\vec{T1}, \vec{T2}, \vec{N})$ coordinate frame where $\vec{T1}, \vec{T2}$ denote principal directions and $\vec{N}$ is the normal.

12.1.3 3D Segmentation from 3D bubbles

An alternative approach is to consider 3D bubbles as evolving *surfaces* guided by the 3D image to capture the geometry of 3D structures. Consider a surface represented by $\psi_0(\xi, \eta) = (x_0(\xi, \eta), y_0(\xi, \eta), z_0(\xi, \eta))$ where ξ and η parameterize the surface, and x_0, y_0, z_0 are Cartesian coordinates where the subscript $_0$ denotes the initial surface prior to deformation. Now, let each point of this surface move by some arbitrary amount in some arbitrary direction, Figure 12.5. This evolution can be described as

$$\begin{cases} \frac{\partial\psi}{\partial t} &= \alpha_1(\xi, \eta, t)\vec{T_1} + \alpha_2(\xi, \eta, t)\vec{T_2} + \beta(\xi, \eta, t)\vec{N} \\ \psi(\xi, \eta, 0) &= \psi_0(\xi, \eta). \end{cases} \qquad \begin{array}{ll} \vec{T_1} = \frac{\frac{\partial\psi}{\partial\xi}}{\left|\frac{\partial\psi}{\partial\xi}\right|} & \text{tangent} \\ \vec{T_2} = \frac{\frac{\partial\psi}{\partial\eta}}{\left|\frac{\partial\psi}{\partial\eta}\right|} & \text{tangent} \end{array} \tag{12.1}$$

By a reparameterization similar to the one described in [112], this can be reduced to an equivalent shape evolution deformation $\frac{\partial\psi}{\partial t} = \beta(\xi, \eta, t)\vec{N}$, $\psi(\xi, \eta, 0) = \psi_0(\xi, \eta)$, where β is again arbitrary, but not necessarily the same as that of the previous equation. Now, concentrate on intrinsic deformations that depend only on the local geometry of the surface at that point, $\frac{\partial\psi}{\partial t} = (\beta_0 - \beta_1\mathcal{K})\vec{\bar{N}}$ where $\mathcal{K}$ a notion of curvature. The intrinsic growth of bubbles is a modulation of this free flow by a stopping term $\frac{\partial\psi}{\partial t} = S(x, y, z)(\beta_0 - \beta_1\mathcal{K})\vec{\bar{N}}$. For theoretical, as well as numerical reasons [154, 61, 106] the evolving surface ψ is captured by an embedding hyper-surface Ψ where $\{\Psi(x, y, z, t) = 0\}$ represent the surface ψ and the evolution is

$$\frac{\partial\Psi}{\partial t} = S(x, y, z)(\beta_0 - \beta_1\mathcal{K})|\nabla\Psi|. \tag{12.2}$$

The stopping term $S(x, y, z)$ is computed at the evolving surface and expanded to all level sets as in [134]. Note that in 3D the straightforward implementation of this term is computationally prohibitive, thus requiring a number of technical revisions to render the simulations practical such as the use of contour-based distance transform [165]; see also the

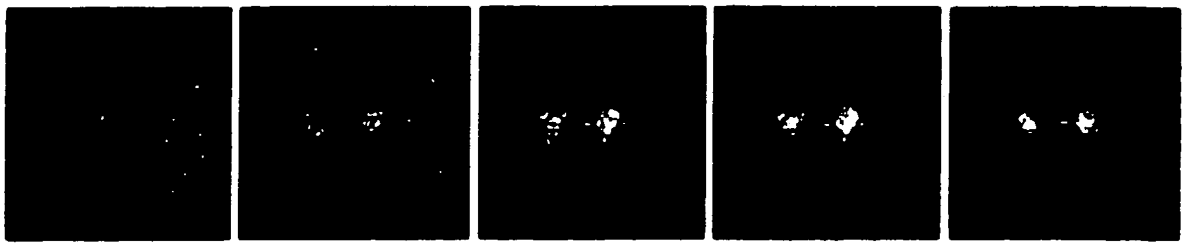


Figure 12.6: The evolution of bubbles as they are initialized on a dumbbell shape, from left to right.

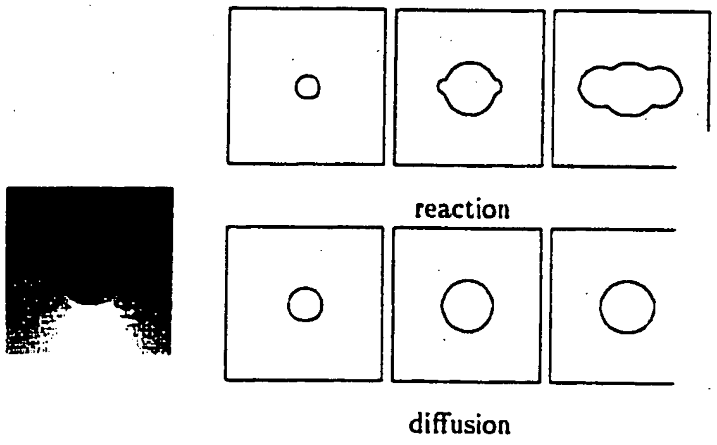
fast-marching scheme [185]. Figure 12.6 illustrates the segmentation of a dumbbell in 3D.

The evolving bubbles can integrate and fuse information provided from different low-level process, *e.g.*, intensity gradients, 3D texture gradients, or from a combinations of imaging modalities, *e.g.*, MR, CT, and SPECT in 3D medical images. This is achieved by a stopping term $S(x, y, z)$ which is a function of the combined gradient in a higher dimensional space, assuming that the two image coordinates are in register.

We now compare this approach to the previous two approaches. Clearly, in contrast to previous approaches, this method is intrinsic and does not rely on a choice of axis. In addition, since 3D bubbles result in a segmented surface, we avoid the reconstruction problem. Finally, the gap problem is dealt with through a notion of diffusion based on a generalization of the curvature-dependent deformation from 2D to 3D, as follows.

12.2 3D Gaps: The Role of Diffusion:

A significant aspect of 2D segmentation by bubbles is the regularization of gaps by the addition of a curvature-dependent deformation, Figure 12.7. The generalization of an appropriate curvature deformation to 3D, however, in confounded by the interaction of two principal curvatures, κ_1 and κ_2. The mean curvature, $\frac{\kappa_1+\kappa_2}{2}$ has been used as a natural extension [98, 35]. However, mean curvature has been shown to split narrow areas in the surface, *e.g.*, a dumbbell, rendering it inappropriate for regularizing across gaps, *e.g.*, a missing piece in a vein, Figure 12.8. Alternative variations were proposed by Whitaker [232]. Neskovic and Kimia [143] studied necessary conditions on the direction of deformation so that it does not lead to self-intersections and found that such deformations must have zero velocity at hyperbolic and parabolic points, and must move elliptic points into their convexity. Secondly, they concluded that the magnitude of deformation must be monotonic and continuous leading t a notion of combined curvature $\mathcal{K} = sign(H)\sqrt{G+|G|}$, referred



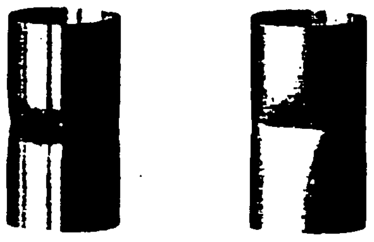


Figure 12.7: This figure illustrates the importance of diffusion in 2D. Top row illustrates a bubble moving with the reaction process and the bottom row illustrates the addition of a diffusion process. In the diffusion process, the bubble does not cross over the gap.

Figure 12.8: This figure illustrates the importance of diffusion in 3D: the evolution by pure reaction (left) penetrates the weak boundary, whereas the evolution with a combined reaction and diffusion (right) does not pass through it.

to as Mean-Gaussian flow leading to the 3D version of the reaction-diffusion space and the flow of 3D bubbles as

$$\frac{\partial \Psi}{\partial t} = S(x,y,z)(\beta_0 - \beta_1 sign(H)\sqrt{G+|G|})|\nabla\Psi|. \qquad (12.3)$$

The mean-Gaussian flow is an appropriate regularization term since elliptic convex points are pushed in, elliptic concave points are push out, but all other points do not influence bubble regularization. A similar result was obtained from the scale analysis of movies [5, 36]. Bubbles display two extremes of behaviors in the 3D reaction-diffusion space: on the one hand, on the reaction side bubbles are "breakable" surfaces that form singularities, capture 3D edges and corners, penetrate narrow straits, and in general are faithful to the minute details of the underlying evidence. On the other hand, on the diffusion side, bubbles are cohesive surfaces that do not easily "break" or form singularities, and as such do not go through small gaps, nor do they respond to small ripples or noise; they integrate local evidence to capture the overall structure.

We use a number of upwind numerical schemes to simulate the 3D bubbles evolution. For example, consider the following implementation of the Osher-Sethian flux [154]:

$$\Psi^{n+1}_{ijk} = \Psi^n_{ijk} - \Delta t S(x,y,z)\beta_0 L_{ijk}(\Psi) - \Delta t S(x,y,z)\beta_1 sign(H)\sqrt{G+|G|}|\nabla\Psi^n_{ijk}| \qquad (12.4)$$

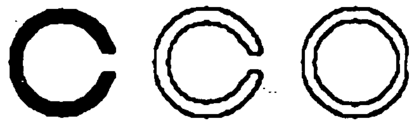
where $Lijk(\Psi) = \sqrt{u^2+v^2+w^2}$ and $u^2 = min^2(D^+_x\Psi, 0) + max^2(D^-_x\Psi, 0)$, *etc.* However, other schemes using Godunouv computations are also possible, leading to accurate sub-pixel implementations as used in [196]. Note that in these computations H is the mean curvature and G is the Gaussian curvature of ψ, as specified below in terms of Ψ

$$H = \frac{\Psi_{xx}(\Psi_y^2 + \Psi_z^2) + \Psi_{yy}(\Psi_x^2 + \Psi_z^2) + \Psi_{zz}(\Psi_x^2 + \Psi_y^2) - 2(\Psi_x\Psi_y\Psi_{xy} + \Psi_y\Psi_z\Psi_{yz} + \Psi_x\Psi_z\Psi_{xz})}{2(\Psi_x^2 + \Psi_y^2 + \Psi_z^2)^{3/2}};$$

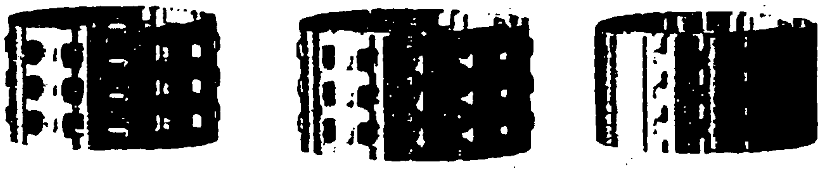
$$G = \frac{\Psi_x^2(\Psi_{yy}\Psi_{zz} - \Psi_{yz}^2) + \Psi_y^2(\Psi_{xx}\Psi_{zz} - \Psi_{xz}^2) + \Psi_z^2(\Psi_{xx}\Psi_{yy} - \Psi_{xy}^2)}{(\Psi_x^2 + \Psi_y^2 + \Psi_z^2)^2} -$$

$$\frac{2[\Psi_x\Psi_y(\Psi_{xz}\Psi_{yz} - \Psi_{xy}\Psi_{zz}) + \Psi_y\Psi_z(\Psi_{xy}\Psi_{xz} - \Psi_{yz}\Psi_{xx}) + \Psi_x\Psi_z(\Psi_{xy}\Psi_{yz} - \Psi_{xz}\Psi_{yy})]}{(\Psi_x^2 + \Psi_y^2 + \Psi_z^2)^2}.$$

Results: We have implemented the 3D bubble model and illustrate its use on synthetic shapes, Figure 12.6 as well as on realistic CT, MRI, and MRA 3D data sets, Figure 12.11. The 3D bubble segmentation technique offers a greater degree of automation as well as the possibility of user interaction, rendering it a powerful tool in a number of medical and other applications.



(a) (b) (c)



(d) (e) (f)

Figure 12.9: Segmentation of a synthetic image of a surface of evolution, a spherical shell with a hole: (TOP) (a) central cross section; (b) reaction, (c) diffusion; (BOTTOM) segmentation of a cylindrical shell with holes: (d) original, (e) reaction, (f) diffusion. Observe that the diffusion process regularizes the gaps in the initial data.

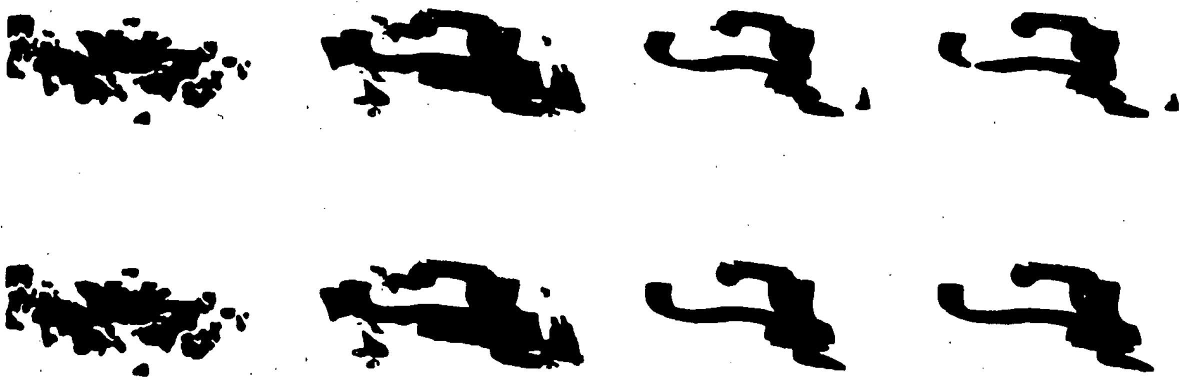


Figure 12.10: A realistic example of a gap in and its regularization on a cropped portion of a 3D MRA image shown in Figure 12.11. The evolution sequence corresponding to reaction (g) and diffusion (h) are shown. Observe how 3D bubbles do not cross over weak gradients in the diffusion process; See Figure 12.11 for full results.

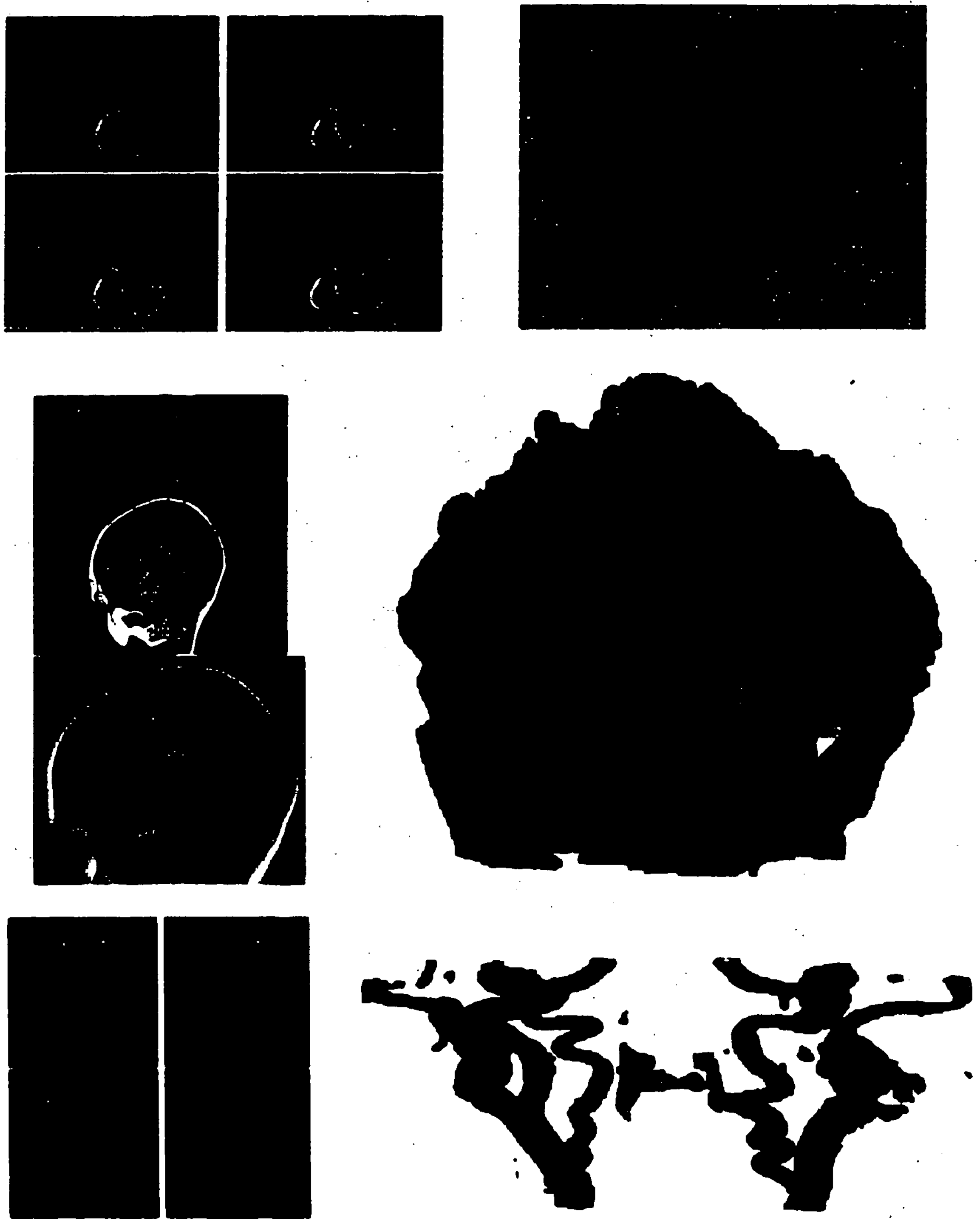


Figure 12.11: The segmentation of 3D images for three different medical applications. (Top) Slices depict carpal bones in a 3D CT image (256x256x64) as reconstructed on the right for joint kinematic studies. 2D slices of 3D bubbles are superimposed on original slices as red boundaries to visually validate the segmentation results. (Middle) The segmentation of whit matter from MRI images of a brain for functional MR studies. (Bott m) The segmentation of carodid arteries for an assessment of atherosclerotic disease.

Chapter 13

Skeletally Coupled Deformable Models For Image Segmentation

This shock-based reaction-diffusion bubbles technique works well when object boundaries have moderate intensity gradients, even when small gaps are present [92]. However, the convergence of bubbles remains unresolved under certain conditions involving weak or diffuse boundaries, large gaps, edges homogeneous only one side, and very narrow regions. This is due to the monotonic nature of growth: once a region has evolved beyond object boundaries it can no longer return to capture it., Figure 13.1.

Our proposed approach is a combination of three of the approaches: Curve evolution [214], seeded region growing [1], and region competition [242]. The approach views the inter-region skeleton as a prediction of final, converged boundaries of growing seeds and feeds back the local statistical desirability of this prediction to the deformation process. In this way, it combines local and global competition under one framework. In the *skeletally coupled deformable models* (SCDM) [177], the inter-region skeleton, the predicted point of collision of the two regions, is used to couple the movements of the two boundaries, *before* they become spatially adjacent. This "long range coupling" does not allow seeds to lose "character" as in region competition, and "symmetrizes" the initialized seeds. To avoid the discretization drawbacks of region competition, and to allow for subpixel movement of the boundaries, SCDM is implemented in the curve evolution framework. This section briefly describes the long range coupling via the skeleton and the subpixel implementation of SCDM.

T see how long range coupling is mediated by the inter-region skelet n, consider Fig-

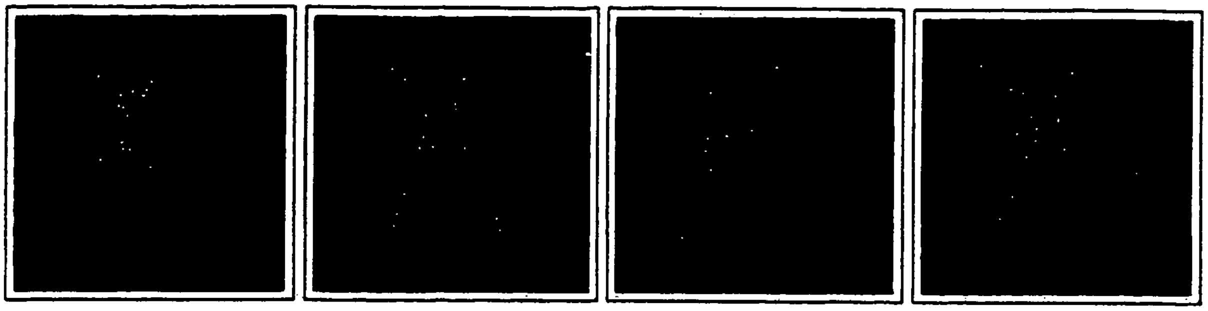


Figure 13.1: Asymmetric initialization of bubbles leads to a cross-over at weak boundaries. thus motivating the ideas of active shocks.

ure 13.2. Let R^+ and R^- be the regions and R_B be the background. Let S denote the *shocks*[1] of R_B as defined in [106, 112, 196, 218]. Consider a point $A^- \in \partial R^-$. the boundary of R^-, and its corresponding shock point $S(A^-) = A$. and let A^+ be such that $S(A^+) = S(A^-) = A$, where skeletal point A is viewed as coupling the deformable model's boundary points, A^+ and A^-. Let the local statistical growth force at a point P be denoted by f_P, typically $f_P = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(I(x,y)-\mu)^2}{2\sigma^2}}$, where $I(x,y)$ is the intensity at P and μ and σ are the mean and standard deviation of the region. The net force at A^- is defined as the sum of competition between local forces f_{A^+} and f_{A^-}, as well as between the forces at the predicted point where the regions would become adjacent. Formally, the first coupling is represented by $(f_{A^+} - \lambda f_{A^-})$ and the second coupling by $(1-\lambda)(f^+ - f^-)$ where λ is a monotonically decreasing function of the distance between A^- and its competitor A^+, captured by $d(A^-, A^+)$, leading to

$$\begin{aligned} F_{A^-} &= f_{A^-} - \lambda f_{A^+} + (1-\lambda)(f^- - f^+) \\ \lambda &= \frac{1}{\sqrt{2\pi\sigma_d^2}} e^{-\frac{d(A^-,A^+)^2}{2\sigma_d^2}} \end{aligned} \tag{13.1}$$

The *local competition* (between local forces f_{A^+} and f_{A^-} on the boundaries of R^+ and R^-) controls the movement of A^- when the regions are close to each other. When the two regions are adjacent, $\lambda = 1$, and the net force is $F_{A^-} = f_{A^-} - f_{A^+}$ as in region competition. On the other hand, the *long range, predicted competition* between the statistic force f^- of A belonging to R^- and f^+ of A belonging to R^+ modulates the movement of A^- when the regions are far away. In Figure 13.2, this long range force coupling has the effect of slowing A^- down, allowing A^+ to "catch up", thus "symmetrizing" the regions R^+ and R^- with respect t the edge.

[1]Sh cks are the skeleton points augmented with the notions of speed, direction, type, label, grouping, and a hierarchy of these groups.

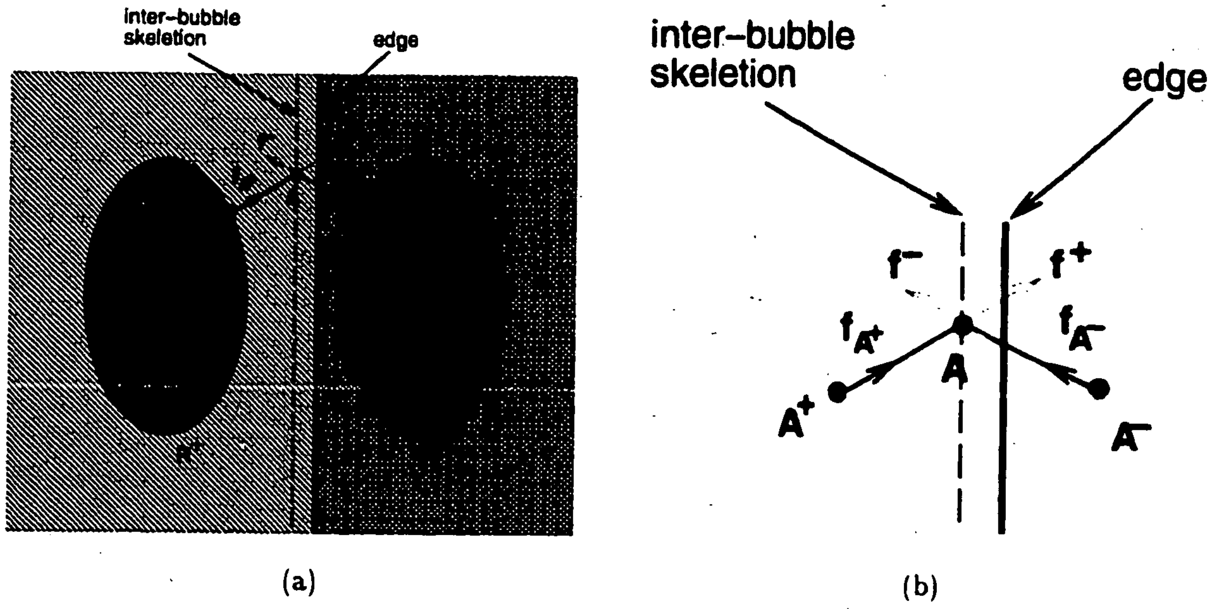
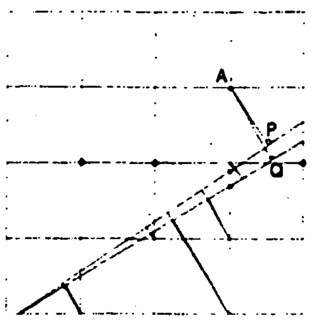


Figure 13.2: A sketch of the coupling of forces between nearby regions, R^+ and R^-, *e.g.*, coupling of A^+ and A^- through A. f_{A^-} and f_{A^+} are the *local* forces at the boundary points A^+ and A^-, respectively. f^- and f^+ are the *predicted* forces at the shock point A due to regions R^+ and R^-. The net force at A^- is computed based on these four forces, Equation 13.1.

This approach is implemented in the curve evolution framework by embedding the curve as the zero level-set of an evolving surface. This allows for partial movement of the curve on a discrete grid. A subpixel implementation of region competition requires regions to be adjacent, thus constantly operating in a mode where more than one curve is present within a pixel. This requires the reliable identification of the subpixel boundary and computation of the forces at subcell points. For the subpixel boundary within a pixel, we use a piecewise circular (PC) approximation [198], which can be extracted accurately from the distance transform using an approach based on Essentially Non Oscillatory (ENO) interpolation [198]. The accuracy of the PC reconstruction from the distance transform, implies that the surface should be transformed in such a way that the modified surface is the distance transform of the modified curve. Hence, the surface point at A in Figure 13.3(a) should be updated by the change from AP to AQ, which for small movements can be approximated by PQ. This requires computing the force at subcell point point P, which requires he accurate computation of the image intensity and image derivatives along the normal to the curve at P. This is done using an ENO interpolation along the line orthogonal t the curve, along the subpixel grid samples $\{A_i\}$ in Figure 13.3(b), which in turn is found using an ENO interpolation al ng grid lines.

(a) 

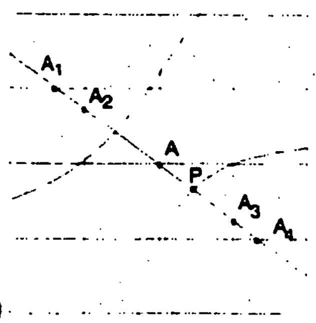
(b) 

Figure 13.3: The subpixel deformation of a curve via updating its embedding surface by the movement of the closest point to each grid point. (a) The subpixel computation of force at P requires a two stage ENO interpolation; one along the line normal to the curve $\{A_i\}$ and one for each A_i along either the horizontal or vertical directions (solid dot). Details are omitted here due to space constraints.

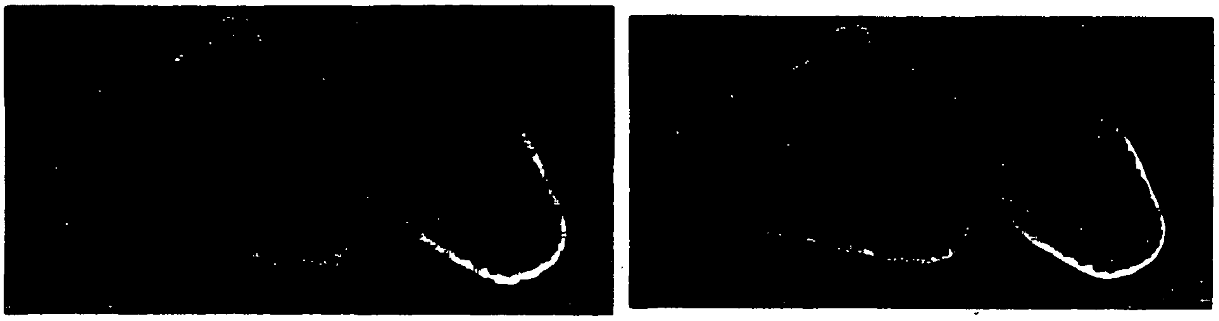
(a) 

Figure 13.4: An example of carpal bone segmentation using skeletally coupled deformable models.

SCDM works well in images having a gap in the bone contour and captures low contrast contours, Figure 13.4. In general, the application of this method to carpal bone segmentation has led to reliable and robust results.

Appendix A

Analytic Distance Functions for Geometric Boundary Models

In Chapter 3 it was shown that the solution to the Eikonal equation, $\nabla u = 1$ constructs the distance transform of the initial curve. In addition, we showed that if the initial source is a point at $P(x_0, y_0)$ with an initial value, $u(x_0, y_0)$, then the wavefront propagates as circular rings, *i.e.*,

$$u(x, y)^2 = (x - x_0)^2 + (y - y_0)^2 \tag{A.1}$$

In this chapter, we present the explicit distance functions of geometric boundary models, *e.g.*, line segments and circular arcs. Figure 3.6 in Chapter 3 illustrates that point sources such as isolated points or end points of boundary models propagate rarefaction waves (lighter regions) and boundary models propagate regular waves (darker regions). Since each geometric model (except singular point source) generates two rarefaction waves from their endpoint and regular waves from the regular portion, we will use the following definitions to determine whether a point on a regular wavefront or a rarefaction wavefront so that correct distance function can be computed.

Definition 12 *(Foot Point): A point P has a* foot *point F on a curve C(s) if dist(P, F) = dist(P, C(s)). Thus, F is a point of C(s) closest to P and with dist(P, F) ≤ dist(P, C(s)) for all s ∈ I.*

Definition 13 *The projective displacement t_P of a point $P(x, y)$ on a geometric boundary model is the signed distance from the foot point of the point P to the beginning point of the geometric boundary model.*

For example, the projective displacement of a point $P(x, y)$ on a line segment AB with end points $A = (x_A, y_A)$ and $B = (x_B, y_B)$ is

$$t_P = \frac{(x - x_A)(x_B - x_A) + (y - y_A)(y_B - y_A)}{(x_B - x_A)^2 + (y_B - y_A)^2} \tag{A.2}$$

where A is assumed to be the beginning point of the line segment AB. Using this definition, it is easy to determine the foot point F of P, *i.e.*,

$$F = (x_F, y_F) = \begin{cases} A = (x_A, y_A) & \text{if } \; t_P < 0 \\ B = (x_B, y_B) & \text{if } \; t_P > 0 \\ A + t_P.AB = (x_A, y_A) + t_P((x_B - x_A), (y_B - y_A)) & \text{if } \; 0 \le t_P \le 1 \end{cases} \tag{A.3}$$

Similarly, the projective displacement of a point $P(x, y)$ on a circular arc segment AB with end points $A = (x_A, y_A)$ and $B = (x_B, y_B)$, and with a center $C = (x_C, y_C)$, and with a radius R is

$$t_P = \begin{cases} \frac{CWT(\theta_A, \theta_F)}{CWT(\theta_A, \theta_B)} & \text{if } \; CWT(\theta_A, \theta_B) > CWT(\theta_A, \theta_F) \\ -\frac{CWT(\theta_A, \theta_F)}{CCWT(\theta_A, \theta_B)} & \text{else if } \; CCWT(\theta_A, \theta_F) < CWT(\theta_B, \theta_F) \\ \frac{CWT(\theta_A, \theta_F)}{CCWT(\theta_A, \theta_B)} & \text{else if } \; CCWT(\theta_A, \theta_F) > CWT(\theta_B, \theta_F) \end{cases} \tag{A.4}$$

where $\theta_F = arctan\frac{y - y_C}{x - x_c}$, and a clock-wise turn from A to B is assumed to determine the circular arc segment. In addition, $CWT(\theta_x, \theta_y)$ specifies clock-wise turn from the angle θ_x to θ_y, and similarly $CCWT$ specifies counter-clock-wise turn in above equation. Analogous to the line case, the foot point F of P on the circular arc segment AB is

$$F = (x_F, y_F) = \begin{cases} A = (x_A, y_A) & \text{if } \; t_P < 0 \\ B = (x_B, y_B) & \text{if } \; t_P > 0 \\ (x_C, y_C) + R(cos(\theta_F), sin(\theta_F)) & \text{if } \; 0 \le t_P \le 1 \end{cases} \tag{A.5}$$

Let us now present the analytic distance function of the wavefront surface, $u(x, y)$ at point $P = (x, y)$:

Definition 14 *The distance between a point $P = (x_P, y_P)$ and a regular parametric curve $C = (x(s), y(s))$ defined on the interval I is given by*

$$dist(P, C(s)) = inf_{s \in I} d(P, C(s)) \tag{A.6}$$

The following proposition from Farouki and Johnstone [65] concludes that the distance (A.6) is satisfied when the $(P - F)$ is perpendicular to the curve $C(s)$. Specifically, the distance can be analytically computed as a result of this proposition:

Proposition 2 *Consider the polynomial $P_\perp$ for the point P and the regular parametric curve $C(s)$*

$$P_\perp = (P - C(s))T(s) = [x_P - x(s)]x'(s) + [y_P - y(s)]y'(s) \tag{A.7}$$

where $\vec{T}(s)$ is the tangent vector at $C(s)$, and let $\{s_1, ..s_N\}$ be the set of odd-multiplicity roots of the polynomial of $P_\perp$ of degree $2N - 1$ on the interior of the interval I. Then the distance function (A.6) can be expressed as

$$dist(P, C(s)) = min_{1 \leq k \leq N} |(P - C(s_k)| \tag{A.8}$$

Proof: See [65] for the full proof and the concluding remarks. In this chapter, we consider only the boundary models of a point, a line segment, and a circular arc segment but the treatment can be extended without loss of generality.

Point: A point $A = (x_A, y_A)$ emanates rarefaction waves as circular rings. The solution $u(x, y)$ is given by

$$u(x, y) = \sqrt{x^2 + y^2 - 2x_A x - 2y_A y + x_A^2 + y_A^2} \tag{A.9}$$

Line Segment: The distance of a point $P(x, y)$ from a line segment between $A = (x_A, y_A)$, and $B = (x_B, y_B)$, Figure 3.6b,

$$u(x, y) = \begin{cases} \sqrt{x^2 + y^2 - 2x_A x - 2y_A y + |\mathbf{A}|^2} & \text{if } t_P < 0 \\ \sqrt{x^2 + y^2 - 2x_B x - 2y_B y + |\mathbf{B}|^2} & \text{else if } t_P > 1 \\ (\frac{(y_B - y_A)}{|\mathbf{AB}|}x - \frac{(x_B - x_A)}{|\mathbf{AB}|}y + \frac{(y_A(x_B - x_A) - x_A(y_B - y_A))}{|\mathbf{AB}|} & \text{else if } 0 \leq t_P \leq 1 \end{cases} \tag{A.10}$$

Circular Arc Segment: The analytic distance function at a point $P = (x, y)$ from a circular arc segment with a center $C = (x_C, y_C)$ and radius, R, Figure 3.6c, is

$$u(x, y) = \begin{cases} \sqrt{x^2 + y^2 - 2x_A x - 2y_A y + |\mathbf{A}|^2} & \text{if } t_P < 0 \\ \sqrt{x^2 + y^2 - 2x_B x - 2y_B y + |\mathbf{B}|^2} & \text{if } t_P > 1 \\ \sqrt{x^2 + y^2 - 2x_C x - 2y_C y + |\mathbf{C}|^2} - R & \text{if } 0 \leq t_P \leq 1 \text{ and } |PC| > R \\ -\sqrt{x^2 + y^2 - 2x_C x - 2y_C y + |\mathbf{C}|^2} + R & \text{if } 0 \leq t_P \leq 1 \text{ and } |PC| < R \end{cases} \tag{A.11}$$

Appendix B

Bisector Curves

Bisector curve for a pair of any combination of point. line segment and circular arc are described below:

Point-Point Bisector: The bisector curve. $b_{pp}(x, y)$ between two points $A = (x_A, y_A)$ and $B = (x_B, y_B)$. $A \neq B$. is given by

$$ax + by + c = 0. \tag{B.1}$$

where $a = 2(x_A - x_B)$, $b = 2(y_A - y_B)$. and $c = (|B|^2 - |A|^2)$.

Point-Line Bisector: The bisector curve, $b_{pl}(x, y)$ between a point $P = (x_P, y_P)$ and the line on which the line segment AB with end points $A = (y_A, y_A)$ and $B = (x_B, y_B)$ lies is given by

$$ax^2 + by^2 + cyx + dx + ey + f = 0, \tag{B.2}$$

where

$$\begin{aligned}
a &= (x_B - x_A)^2, \\
b &= (y_B - y_A)^2, \\
c &= 2(x_B - x_A)(y_B - y_A), \\
d &= -2x_P|AB|^2 - 2(y_B - y_A)(y_A(x_B - x_A) - x_A(y_B - y_A)), \\
e &= -2y_P|AB|^2 + 2(x_B - x_A)(y_A(x_B - x_A) - x_A(y_B - y_A)), \\
f &= |P|^2|AB|^2 - (y_A(x_B - x_A) - x_A(y_B - y_A))^2.
\end{aligned} \tag{B.3}$$

Point-Arc Bisect r: The bisector curve, $b_{pc}(x, y)$ between a point $P = (x_P, y_P)$ and a circular arc segment with a center $C = (x_C, y_C)$ and radius R is given by

$$ax^2 + by^2 + cyx + dx + ey + f = 0, \tag{B.4}$$

where

$$\begin{aligned} a &= 4(x_P - x_C)^2 - 4R^2 \\ b &= 4(y_P - y_C)^2 - 4R^2 \\ c &= 8(x_P - x_C)(y_P - y_C) \\ d &= -4(x_P - x_C)(|P|^2 - |C|^2) + 4(x_P + x_C)R^2 \\ e &= -4(y_P - y_C)(|P|^2 - |C|^2) + 4(y_P + y_C)R^2 \\ f &= (|P|^2 - |C|^2)^2 - 2(|P|^2 + |C|^2)R^2 + R^4 \end{aligned} \tag{B.5}$$

Line-Line Bisector: The bisector curve, $b_{ll}(x, y)$ between the lines on which the line segment AB with end points $A = (y_A, y_A)$ and $B = (x_B, y_B)$, $A \neq B$, and a line segment CD with end points $C = (y_C, y_C)$ and $D = (x_D, y_D)$, $C \neq D$, lie is given by

$$ax + by + c = 0, \tag{B.6}$$

where

$$\begin{aligned} a &= (y_B - y_A)|CD|n_1 - (y_D - y_C)|AB|n_2, \\ b &= -(x_B - x_A)|CD|n_1 + (x_D - x_C)|AB|n_2, \\ c &= (y_A(x_B - xA) - x_A(y_B - y_A))|CD|n_1 - (y_C(x_D - x_C) - x_C(y_D - y_C))|AB|n_2. \end{aligned} \tag{B.7}$$

where $|AB| = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$, $|CD| = \sqrt{(x_C - x_D)^2 + (y_C - y_D)^2}$, and $n_1 = \pm 1$ and $n_2 = \pm 1$. Specifically, the distance from a line segment is positive for points which are in the normal direction of the boundary model and negative otherwise. Note that in our wave propagation algorithm each wavefront carries this information, *i.e.*, direction of wave respect to the direction of normal of its boundary model, so that unnecessary computations are avoided.

Arc-Arc Bisector: The bisector curve, $b_{cc}(x, y)$ between a circular arc segment with a center $C = (x_C, y_C)$ and radius R_1, and a circular arc segment with center $D = (x_D, y_D)$ and radius R_2 is given by

$$ax^2 + by^2 + cyx + dx + ey + f = 0, \tag{B.8}$$

where

$$\begin{aligned}
a &= 4(x_D - x_C)^2 - 4(n_2R_2 - n_1R_1)^2\\
b &= 4(y_D - y_C)^2 - 4(n_2R_2 - n_1R_1)^2\\
c &= 8(x_D - x_C)(y_D - y_C)\\
d &= -4(x_D - x_C)(|D|^2 - |C|^2) + 4(x_D + x_C)(n_2R_2 - n_1R_1)^2\\
e &= -4(y_D - y_C)(|D|^2 - |C|^2) + 4(y_D + y_C)(n_2R_2 - n_1R_1)^2\\
f &= (|D|^2 - |C|^2)^2 - 2(|D|^2 + |C|^2)(n_2R_2 - n_1R_1)^2 + (n_2R_2 - n_1R_1)^4
\end{aligned} \tag{B.9}$$

where $n_1 = \pm 1$ and $n_2 = \pm 1$.

Line-Arc Bisector: The bisector curve, $b_{pl}(x, y)$ between a line segment AB with end points $A = (y_A, y_A)$ and $B = (x_B, y_B)$, $A \neq B$ and a circular arc segment with center $C = (x_C, y_C)$ and radius R is given by

$$ax^2 + by^2 + cyx + dx + ey + f = 0. \tag{B.10}$$

where

$$\begin{aligned}
a &= (x_B - x_A)^2\\
b &= (y_B - y_A)^2\\
c &= 2(x_B - x_A)(y_B - y_A)\\
d &= -2x_C|AB|^2 - 2(y_B - y_A)((y_A(x_B - x_A) - x_A(y_B - y_A))n_1 - R|AB|^2n_2)\\
e &= -2y_C|AB|^2 + 2(x_B - x_A)((y_A(x_B - x_A) - x_A(y_B - y_A))n_1 - R|AB|^2n_2)\\
f &= |C|^2|AB|^2 - ((y_A(x_B - x_A) - x_A(y_B - y_A))n_1 - R|AB|^2n2)^2
\end{aligned} \tag{B.11}$$

where $n_1 = \pm 1$ and $n_2 = \pm 1$.

The bisector for these geometric boundary models can be computed exactly. Figure B.1 illustrates an example for each of these cases. In addition, the bisector between the conic, cubic, or higher degree polynomial may be approximated by numerical methods [166]

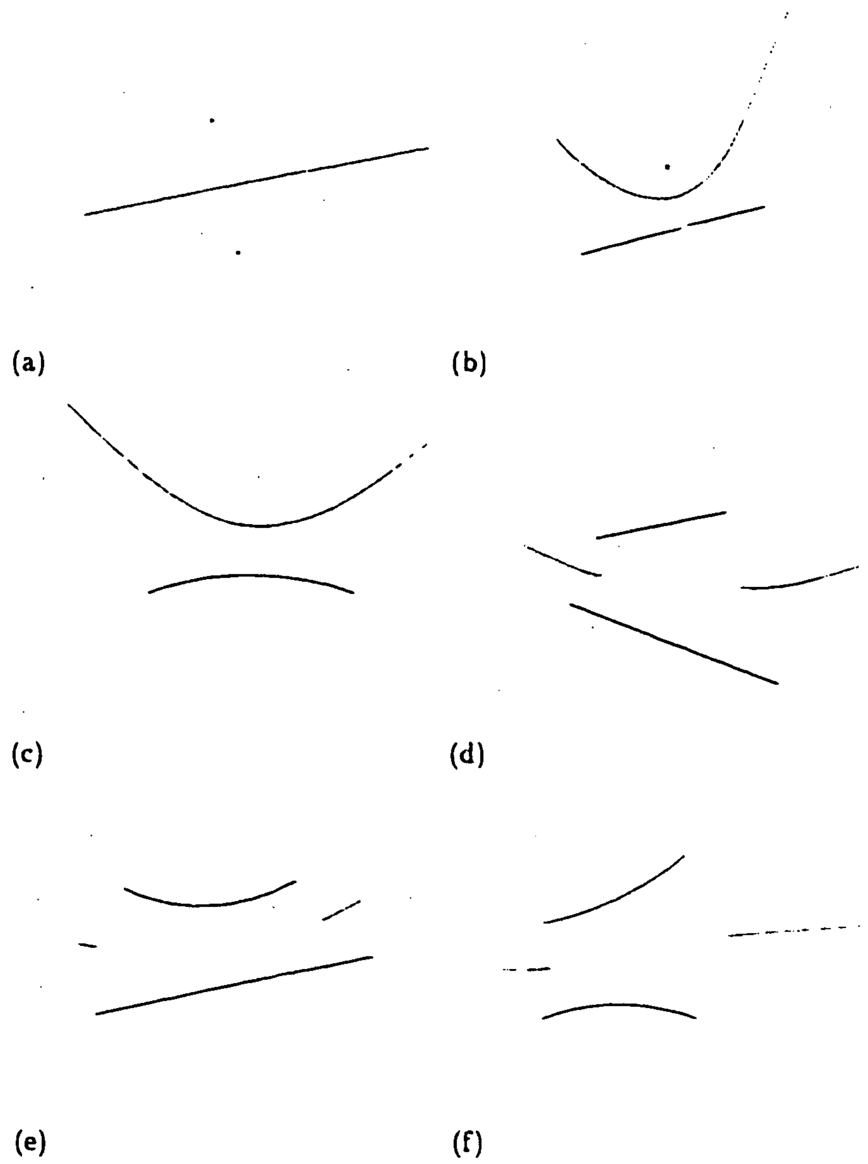


Figure B.1: This figure illustrates the bisector curve between (a) a point and a point, (b) a point and a line (c) a circular arc and a point (d) a line and a line (e) a line and a circular arc (f) two circular arc segment. Note that the boundary models are bounded and the true bisector curve is given, *i.e.*, bisectors from the end points are also included.

Appendix C

Second-Order Shocks

Let us now specifically present the second-order shock computation for a number of geometric boundary models, namely, *point*, *line*, and *circular arc*.

Point-Point second-order shocks: The second-order shock location (x, y) from two point sources, $A = (x_A, y_A)$ and $B = (x_B, y_B))$ is given as (Figure C.1a

$$\begin{aligned} x &= \tfrac{(x_B + x_A)}{2}, \\ y &= \tfrac{(y_B + y_A)}{2}. \end{aligned} \tag{C.1}$$

Point-Line second-order shocks: The second-order shock location (x, y) from a point $C = (x_C, y_C)$ and a line segment AB with end points $A = (y_A, y_A)$ and $B = (x_B, y_B)$ is given by (Figure C.1b)

$$\begin{aligned} x &= \tfrac{(x_C + x_F)}{2} \\ y &= \tfrac{(y_C + y_F)}{2} \end{aligned} \tag{C.2}$$

where $F = (x_F, y_F)$ is the foot point. The coordinates of the foot point are on the line segment

$$\begin{aligned} x_F &= x_A + t_C(x_B - x_A) \\ x_F &= y_A + t_C(y_B - y_A), \end{aligned} \tag{C.3}$$

and $CF \perp AB$, thus

$$t_C = \frac{\vec{CA}.\vec{AB}}{|AB|^2} \tag{C.4}$$

with the requirement that $0 \leq t_C \leq 1$ so that F belongs to the line segment AB.

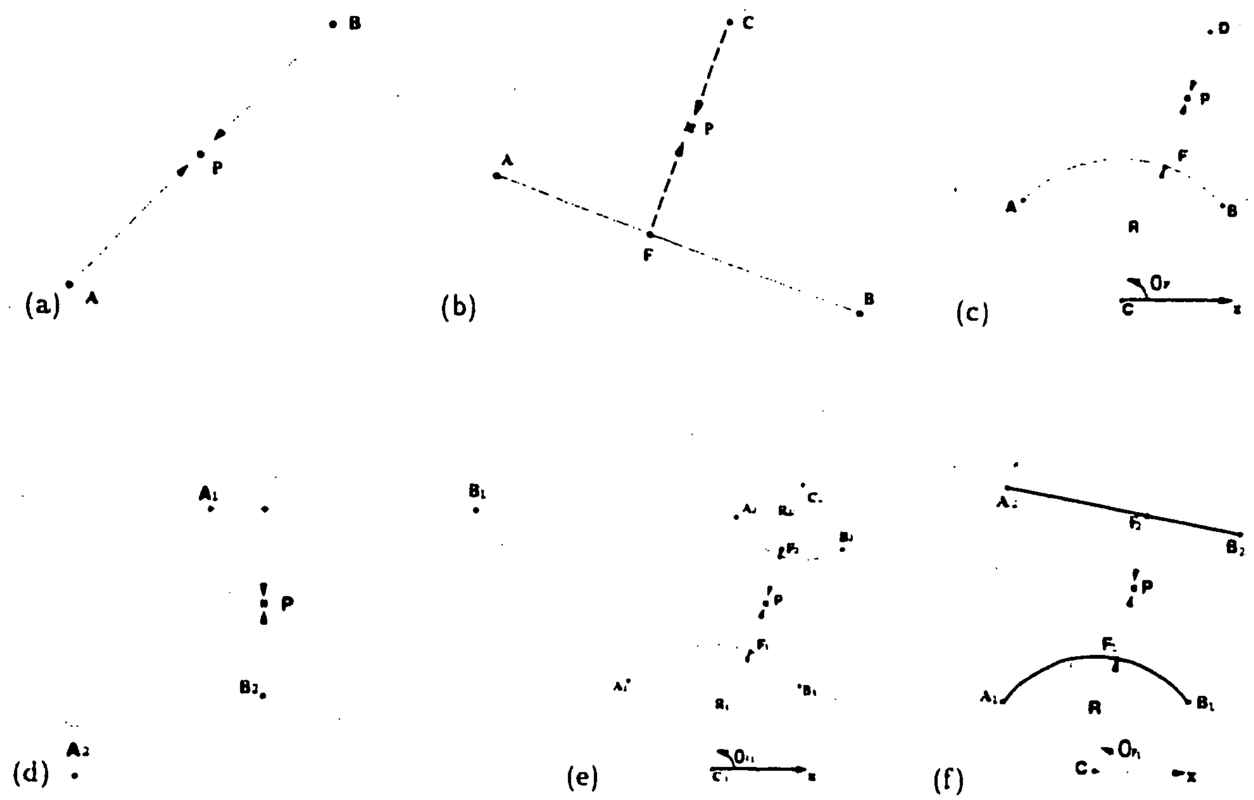


Figure C.1: Second Order Shock from point and (a) point, (b) line and (c) arc segment, (d) a line and line, (e) line and arc and (f) arc and arc segment.

Point-Arc second-order shocks: The second order shock location (x, y) from a point $D = (x_D, y_D)$ and a circular arc segment AB with end points $A = (y_A, y_A)$ and $B = (x_B, y_B)$, and a center $C = (x_C, y_C)$ and radius, R is given by (Figure C.1c)

$$\begin{aligned} x &= x_C + |PC|\cos(\theta) = x_C + |PC|\tfrac{(x_D - x_C)}{|DC|} \\ y &= y_C + |PC|\sin(\theta) = y_C + |PC|\tfrac{(y_D - y_C)}{|DC|} \end{aligned} \qquad \text{(C.5)}$$

Observe that if $R > |DC|$, then $|PC| = R + \frac{|DF|}{2}$ and $|DF| = R - |DC|$, thus $|PC| = \frac{R+|DC|}{2}$. If $R < |DC|$, then $|PC| = R - \frac{|DF|}{2}$ and $|DF| = |DC| - R$, thus $|PC| = \frac{R+|DC|}{2}$. Then we have

$$\begin{aligned} x &= x_C + \tfrac{(R+|DC|)(x_D - x_C)}{2|DC|} \\ y &= y_C + \tfrac{(R+|DC|)(y_D - y_C)}{2|DC|} \end{aligned} \qquad \text{(C.6)}$$

with the requirement that $0 \leq t_C \leq 1$ so that F belongs to the circular arc segment AB; see Appendix A for the computation of the projective displacement t_C of a point (x, y) on a circular arc.

Line-Line second-order shocks: Second-order shocks from two lines can always be detected from an end point of a line and a line. Thus, a second order shock computation from two line segments reduces to "Point-Point" or "Point-Line" case, Figure C.1d.

Arc-Arc sec nd order shocks: Let us consider the second order shock location (x, y) from a circular arc segment A_1B_1 with end points $A_1 = (x_{A_1}, y_{A_1})$ and $B_1 = (x_{B_1}, y_{B_1})$ and with a center $C_1 = (x_{C_1}, y_{C_1})$ and radius, R_1 and a circular arc segment A_2B_2 with end points $A_2 = (x_{A_2}, y_{A_2})$ and $B_2 = (x_{B_2}, y_{B_2})$ and with a center $C_2 = (x_{C_2}, y_{C_2})$ and radius. R_2. We need to consider only the case when $|C_2C_1| > (R_1 + R_2)$, Figure C.1e since the other cases reduce to "Point-Point" or "Point-Arc" case. The second order shock location (x, y) is

$$\begin{aligned} x &= x_{C_1} + |PC_1|cos(\theta) = x_{C_1} + |PC_1|\tfrac{(x_{C_2}-x_{C_1})}{|C_1C_2|} \\ y &= y_{C_1} + |PC_1|sin(\theta) = y_{C_1} + |PC_1|\tfrac{(y_{C_2}-y_{C_1})}{|C_1C_2|} \end{aligned} \tag{C.7}$$

Noting that $|PC_1| = R_1 + \frac{|F_1F_2|}{2}$ and $|F_1F_2| = |C_2C_1| - (R_1 + R_2)$, we have $|PC_1| = \frac{(R_1+|C_1C_2|-R_2)}{2}$. Inserting this into the equation above we have

$$\begin{aligned} x &= x_{C_1} + \tfrac{(R_1+|C_1C_2|-R_2)(x_{C_2}-x_{C_1})}{2|C_1C_2|} \\ y &= y_{C_1} + \tfrac{(R_1+|C_1C_2|-R_2)(y_{C_2}-y_{C_1})}{2|C_1C_2|} \end{aligned} \tag{C.8}$$

with the requirements that $0 \le t_{C_1} \le 1$ and $0 \le t_{C_2} \le 1$ so that F_1 and F_2 belong to the circular arc segment A_1B_1 and A_2B_2, respectively; see Appendix A for the computation of the projective displacement t_{C_1} (t_{C_2}) of a point (x, y) on a circular arc.

Arc-Line second-order shocks: Let us now consider the second order shock location (x, y) from a circular arc segment A_1B_1 with end points $A_1 = (y_{A_1}, y_{A_1})$ and $B_1 = (x_{B_1}, y_{B_1})$. and with a center $C = (x_C, y_C)$ and radius, R and a line segment A_2B_2 with end points $A_2 = (y_{A_2}, y_{A_2})$ and $B_2 = (x_{B_2}, y_{B_2})$. We need to consider only the case when $min(|CA_2|, |CB_2|) > R$, (Figure C.1f) since the other cases reduce to "Point-Point" and "Point-Arc" cases. The second order shock location (x, y) is

$$\begin{aligned} x &= x_C + |PC|cos(\theta) = x_C + |PC|\tfrac{(x_{F_2}-x_C)}{|CF_1|} \\ y &= y_C + |PC|sin(\theta) = y_C + |PC|\tfrac{(y_{F_2}-y_C)}{|CF_1|} \end{aligned} \tag{C.9}$$

where

$$\begin{aligned} |PC| &= R + \tfrac{|F_1F_2|}{2} = R + \tfrac{R+|CF_2|}{2} \\ x_{F_2} &= x_{A_2} + t_C(x_{B_2} - x_{A_2}) \\ x_{F_2} &= y_{A_2} + t_C(y_{B_2} - y_{A_2}) \\ t_C &= \tfrac{\vec{CA_2} \cdot \vec{A_2B_2}}{|A_2B_2|^2} \end{aligned} \tag{C.10}$$

with the requirements that $0 \leq t_C \leq 1$ and $0 \leq t_{F_1} \leq 1$ so that F_1 and F_2 belong to the circular segment A_1B_1 and the line arc segment A_2B_2, respectively; see Appendix A for the computation of the projective displacement t_C of a point (x, y) on a circular arc.

Bibliography

[1] R. Adams and L. Bischof. Seeded region growing. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 16(6):641–647, 1994.

[2] N. Ahuja and M. Tuceryan. Extraction of early perceptual structure in dot patterns: integrating region, boundary, and component gestalt. *Computer vision, Graphics, and Image Processing*, 48:304–356, 1989.

[3] H. Alt and O. Schwarzkopf. The Voronoi Diagram of curved objects. In *ACM Symp. on Computational Geometry*, 1995.

[4] T. Alter and R. Basri. Extracting salient curves from images: An analysis of the saliency network. *IJCV*, 27(1):51–69, March 1998.

[5] L. Alvarez, F. Guichard, P.-L. Lions, and J.-M. Morel. Axioms and fundamental equations of image processing. *Arch. Rational Mech. Anal.*, 123:199–257, 1993.

[6] L. Alvarez, P.-L. Lions, and J.-M. Morel. Image selective smoothing and edge detection by nonlinear diffusion: II. *SIAM Journal of Numerical Analysis*, 29(3):845–866, June 1992.

[7] L. Alvarez and J.-M. Morel. Formalization and computational aspects of image analysis. Technical Report 0493, Departmento de Informatica y Sistemas, Universidad de Las Palmas de Gran Canaria, October 1993.

[8] A. A. Amini, T. Weymouth, and R. Jain. Using dynamic programming for solving variational problems in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:855–867, 1990.

[9] C. Arcelli and G. Sanniti di Baja. A width-independent fast thinning algorithm. *IEEE Trans. on PAMI*, PAMI-7(4):463–474, July 1985.

[10] C. Arcelli and G. Sanniti di Baja. Euclidean skeleton via centre-of-maximal disc extraction. *Image and Vision Computing*, 11(3):163–173, 1993.

[11] A. Ar hart, L. Vincent, and B. B. Kimia. Mathematical morphology: The Hamilton-Jacobi connection. Technical Report LEMS 108, Brown University, June 1992.

[12] A. Arehart, L. Vincent, and B. B. Kimia. Mathematical morphol gy: The Hamilton-Jacobi connection. In *Proceedings of the Fourth International Conference on Computer Vision (Berlin, Germany, May 11-13, 1993)*, pages 215–219, Washington, DC, 1993. IEEE Computer Society Press.

[13] H. Asada and M. Brady. The curvature primal sketch. *IEEE PAMI*, 8:2–14, 1983.

[14] D. Atalli, G. S. diBaja, and E. Thiel. Pruning discrete and semi-discontinous skeletons. In *Proc. of 8th ICAP*, San Remo, September 1995.

[15] J. August, K. Siddiqi, and S. W. Zucker. Fragment grouping via the principle of perceptual occlusion. In *International Conference on Pattern Recognition*, pages 1–8, 1996.

[16] F. Aurenhammer. Voronoi diagrams - a survey of fundamental geometric data structure. *ACM Computing Survey*, 23(3):345–405, 1991.

[17] J. Babaud, A. P. Witkin, M. Baudin, and R. O. Duda. Uniqueness of the gaussian kernel for scale-space filtering. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(1):26–33, January 1986.

[18] R. Bajcsy and S. Kovačič. Multiresolution elastic matching. *Computer Vision, Graphics and Image Processing*, 46:1–21, 1989.

[19] D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice Hall, New Jersey, 1982.

[20] G. Barles. Remarks on a flame propagation model. Technical Report No 464, INRIA Rapports de Recherche, December 1985.

[21] M. Barzohar and D. Cooper. Automatic finding of main roads in aerial images by using geometric-stochastic models and estimation. *PAMI*, 18(7):707–721, July 1996.

[22] M. O. Berger and R. Mohr. Towards autonomy in active contour models. In *Proceedings of the Tenth International Conference on Pattern Recognition*, pages 847–851, 1990.

[23] N. Bleistein. *Mathematical Methods For Wave Phenomena*. Academic Press, Inc, Orlanda, Florida, 1984.

[24] H. Blum. Biological shape and visual science. *J. Theor. Biol.*, 38:205–287, 1973.

[25] H. Blum and R. N. Nagel. Shape description using weighted symmetric axis features. *Pattern Recognition*, 10:167–180, 1978.

[26] G. Borgefors. Distance transformations in arbitrary dimensions. *CVGIP*, 27:321–345, 1984.

[27] M. Born and E. Wolf. *Principles of Optics*. Pergamon Press, 1959.

[28] J. Brandt and V. Algzai. Continuous skeleton computation by voronoi diagram. *CVGIP*, 55(3):329–338, 1982.

[29] J. W. Brandt, A. J. Jain, and V. R. Algazi. Medial axis representation and encoding of scanned documents. *Journal of Visual Communications and Image Representation*, 2:329–338, 1991.

[30] R. Brockett and P. Maragos. Evolution equations for continuous-sacle morphology. In *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing*, San Francisco, CA, March 1992.

[31] M. A. Butt and P. Maragos. Optimal design of chmafer distance transforms. *IEEE Trans. Image Processing*, 1998.

[32] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8:679–698, 1986.

[33] V. Caselles, F. Catte, T. Coll, and F. Dibos. A geometric model for active contours in image processing. Technical Report No 9210, CEREMADE, 1992.

[34] V. Caselles, F. Catte, T. Coll, and F. Dibos. A geometric model for active contours in image processing. *Numerische Mathematik*, 66, 1993.

[35] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. In ICCV95 [90], pages 156–162.

[36] V. Caselles and C. Sbert. What is the best causal scale space for 3d images? Technical Report preprint, Departmento de Informatica y Sistemas, Universidad de Las Palmas de Gran Canaria, 1994.

[37] F. Catte, F. Dibos, and G. Koepfler. A morphological scheme for mean curvature motion and applications to anisotropic diffusion and motion of level sets. *SIAM Journal of Numerical Analysis*, 32-6:1895–1909, 1995.

[38] F. Catté, P.-L. Lions, J.-M. Morel, and T. Coll. Image selective smoothing and edge detection by nonlinear diffusion. *SIAM Journal of Numerical Analysis*, 29(1):182–193, February 1992.

[39] A. Chakraborty, L. H. Staib, and J. S. Duncan. Deformable boundary finding influenced by region homogoneity. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 624–627, 1994.

[40] S. Chen and W. Lin. Split-and-merge image segmentation based n the localized feature analysis and statistical tests. *CVGIP: Graphical Models and Image Processing*, 53(5):457–475, 1991.

[41] Y. Chen, Y. Giga, and S. Goto. Uniqueness and existence of viscosity solutions of generalized m an curvature flow equations. *Journal f Differential Geometry*, 33(3):749–786, 1991.

[42] L. P. Chew. Constrained delaunay triangulation. *Algorithmica*, 4:97–108, 1989.

[43] H. I. Choi, S. W. Ch i, H. P. Moon, and N. S. Wee. New algorithm for medial transform of the plane domain. *Graphical Models and Image Processing*, 59:463–483, 1997. 6.

[44] J. J. Chou. Voronoi diagrams for planar shapes. *IEEE Computer Graphics and Applications*, 15(2):52-59, 1995.

[45] F. Cohen and D. Cooper. Simple parallel hierarchical and relaxation algorithms for segmenting noncausal markovian random fields. *IEEE Transations on Pattern Analysis and Machine Intelligence*, pages 195–219, 1987.

[46] L. D. Cohen. On active contour models. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 53:211-218, 1991.

[47] L. D. Cohen and I. Cohen. Using a finite element method for active contour models and 3D reconstruction from cross sections. In *Proc. Third International Conference on Computer Vision*, pages 587–591, 1990.

[48] L. D. Cohen and I. Cohen. Finite element methods for active contour models and balloons for 2D and 3D images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15:1131–1147, 1993.

[49] D. Cooper, H. Elliott, F. Cohen, L. Reiss, and P. Symosek. Stochastic boundary estimation and object recognition. *Comp. Graph. and Image Proc.*, 12:326–356, 1980.

[50] L. P. Cordella and G. S. diBaja. Context dependent smoothing of figures represented by their medial axes. In *Proc. of 8th ICPR*, pages 280–282, 1986.

[51] R. Courant and D. Hilbert. *Methods of Mathematical Physics, II.* Interscience Publisher, 1962.

[52] *CVPR'99 (IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Fort Collins, Colorado, USA, June)*, Fort Collins, Colorado, USA, June 23-25 1999. IEEE Computer Society Press.

[53] P. Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14:227–248, 1980.

[54] C. David and S. W. Zucker. Potentials, valleys, and dynamic global coverings. *International Journal of Computer Vision*, 5:219–238, 1990.

[55] E. DeMicheli, B. Caprile, P. Ottonello, and V. Torre. Localization and noise in edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 11:1106–1117, 1989.

[56] R. V. den Boomgaard. Mathematical morphology: extensions towards computer vision. Ph.D. dissertation, University of Amsterdam, March 1992.

[57] A. R. Dill, M. D. Levine, and P. B. Noble. Multiple resolution skeletons. *IEEE transactions n Pattern Analysis and Machine Intelligence*, 9(4):495–504, July 1987.

[58] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* Wiley, 1973.

[59] H. Eggers. Two fast Euclidean distance transformations in Z^2 based on sufficient propagation. *Computer Vision and Image Understanding*, 69:106–116, 1998.

[60] G. Elber and M.-S. Kim. Bisector curves of freeform planar curves. Technical report, Technion-ComputerScience Department, TR-CIS9707, 1997.

[61] L. C. Evans and J. Spruck. Motion of level sets by mean curvature I. *Journal of Differential Geometry*, 33(3):635–681, May 1991.

[62] L. C. Evans and J. Spruck. Motion of level sets by mean curvature II. *Trans. Amer. Math. Soc.*, 330:321–332, 1992.

[63] L. C. Evans and J. Spruck. Motion of level sets by mean curvature III. *J. Geom. Analysis*, 2:121–150, 1992.

[64] R. Farouki and J. Johnstone. Computing point/curve and curve/curve bisectors. In R. B. Fisher, editor, *Design and Application of Curves and Surfaces: Mathematics of Surfaces*, pages 327–354. Oxford University Press, 1994.

[65] R. T. Farouki and J. K. Johnstone. The bisector of a point and a plane parametric curve. *Computer Aided Geometric Design*, 11:117–151, 1994.

[66] M. Fischler and R. Bolles. Perceptual organization and curve partitioning. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8:100–105, 1986.

[67] P. Fua and Y. Leclerc. Model driven edge detection. *Machine Vision Applications*, 1:45–56, 1990.

[68] M. Gage. Curve shortening makes convex curves circular. *Invent. Math.*, 76:357–364, 1984.

[69] M. Gage. On an area-preserving evolution equation for plane curves. *Contemp. Math.*, 51:51–62, 1986.

[70] M. Gage and R. S. Hamilton. The heat equation shrinking convex plane curves. *J. Differential Geometry*, 23:69–96, 1986.

[71] Y. Ge and J. M. Fitzpatric. On the generation of skeletons from discrete euclidean maps. *IEEE Trans. on PAMI*, 18:1055–1067, 1996.

[72] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transations on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[73] P. J. Giblin and B. B. Kimia. On the local form and transitions of symmetry sets, and medial axes, and shocks in 2D. Technical Report LEMS-170, LEMS, Brown University, April 1998.

[74] P. J. Giblin and B. B. Kimia. On the intrinsic reconstruction of shape from its symmetries. In CVPR1999 [52], pages 79–84.

[75] P. J. Giblin and B. B. Kimia. On the l cal form and transitions of symmetry sets, and medial axes, and shocks in 2D. In ICCV1999 [89], pages 385–391.

[76] B. Gillam. New evidence for "closure" in perception. *Perception and Psychophysics*, 17(5):521–524, 1975.

[77] M. A. Grayson. The heat equation shrinks embedded plane curves to round points. *J. Differential Geometry*, 26:285–314, 1987.

[78] G. Guy and G. Medioni. Inferring global perceptual contours from local features. *International Journal of Computer Vision*, 20(1/2):113–133, 1996.

[79] R. M. Haralick, S. R. Sternberg, and X. Zhuang. Image analysis using mathematical morphology. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 9(4):532–550, July 1987.

[80] A. Harten. High resolution schemes for hyperbolic conservation laws. *Journal of Computational Physics*, 49:357–393, 1983.

[81] M. Held. *On the computational geometry of Pocket Machining*. Springer-Verlag, Berlin, 1991.

[82] L. Herault and R. Horaud. Figure-ground discrimination: A combinatorial approach. *PAMI*, 15(9):899–914, September 1993.

[83] S.-B. Ho and C. R. Dyer. Shape smoothing using medial axis transform. *IEEE Trans. on PAMI*, 8(4):512–520, July 1986.

[84] C. Hoffmann and P. Vermeer. Eliminating extraneous solutions in curve and surface operation. *International Journal of Computational Geometry and Applications*, 1(1):47–66, 1991.

[85] B. Horn and E. Weldon. Filtering closed curves. *IEEE PAMI*, 8(5), 1986.

[86] B. K. P. Horn. The curve of least energy. *ACM Trans. Mathematical Software*, 9(4):442–460, December 1983.

[87] R. Hummel and S. W. Zucker. On the foundations of relaxation labeling processes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 6:267–287, 1983.

[88] *Second International Conference on Computer Vision (Tampa,, FL, December 5-8, 1988)*, Washington, DC,, 1988. IEEE Computer Society Press.

[89] *ICCV'99 (Seveth International Conference on Computer Vision, KerKyra, Greece, September 20-25, 1999*, KerKyra, Greece, September 20-25 1999. IEEE Computer Society Press.

[90] *Fifth International Conference on Computer Vision*, Boston, Massachusetts, Jun 1995. IEEE Computer Society Press.

[91] B.-K. Jang and R. T. Chin. Analysis of thinning algorithms using mathematical morphology. *IEEE Trans. on PAMI*, 12(6):541–551, June 1990.

[92] M. Jayaraman, B. Kimia, H. Tek, G. Tung, and J. Rogg. Semi-automatic image segmentation of primary brain tumors based on deformable bubbles. In *Proceedings of the Radiological Society of North America*, November 1997.

[93] A. Jeffrey and T. Taniuti. *Non-linear Wave Propagation*. Academic Press, 1964.

[94] F. John. *Partial Differential Equations*. Springer-Verlag, 1971.

[95] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *Int. Journal of Computer Vision*, 1:321–331, 1987.

[96] M. F. Kelly and M. D. Levine. Annular symmetry operators: A method for locating and describing objects. In *Fifth International Conference on Computer Vision*, pages 1016–1021, 1995.

[97] J. Kevorkian. *Partial Diffential Equations: Analytic Solution Techniques*. The Wadsworth and Brooks/Cole Mathematics Series, 1990.

[98] S. Kichenassamy, A. Kumar, P. Olver, A. Tannenbaum, and A. Yezzi. Gradient flows and geometric active contour models. In *Fifth International Conference on Computer Vision*, pages 810–815, 1995.

[99] B. B. Kimia. Conservation laws and a theory of shape. Ph.D. dissertation, McGill Center for Intelligent Machines, McGill University, Montreal, Canada, 1990.

[100] B. B. Kimia. Content-based retrieval of images based on shape. In L. Bergman and V. Castelli, editors, *Image Databases, Search and Retrieval of Digital Imagery*. John Wiley and Sons, 1999.

[101] B. B. Kimia, J. Chan, D. Bertrand, S. Coe, Z. Roadhouse, and H. Tek. A shock-based approach for indexing of image databases using shape. In *Proceedings of the SPIE's Multimedia Storage and Archiving Systems II*, volume 3229, pages 288–302, Dallas, Texas, November 1997.

[102] B. B. Kimia, I. Frankel, and A.-M. Popescu. Euler spiral for shape completion. *IEEE Trans. Pattern Analysis and Machine Intelligence*, pages To be Submitted, January, 2000.

[103] B. B. Kimia and K. Siddiqi. Geometric heat equation and nonlinear diffusion of shapes and images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 113–120, Seattle, Washington, June 1994. IEEE Computer Society Press.

[104] B. B. Kimia and K. Siddiqi. Geometric heat equation and nonlinear diffusion of shapes and images. *Computer Vision Graphics and Image Processing: Image Understanding*, 64(3):305–322, November 1996.

[105] B. B. Kimia, A. R. Tannenbaum, and S. W. Zucker. Toward a computational theory of shap : An overview. CIM-89-13, McGill Center for Intelligent Machines, McGill University, Montreal, Canada, 1989.

[106] B. B. Kimia, A. R. Tannenbaum, and S. W. Zucker. Toward a computational theory of shape: An overview. In *Proceedings of the First European Conference on Computer Vision*, pages 402–407, Antibes, France, 1990. Springer Verlag.

[107] B. B. Kimia, A. R. Tannenbaum, and S. W. Zucker. Toward a computational theory of shape: An overview. In O. Faugeras, editor, *Proceedings of the First European Conference on Computer Vision*, pages 402–407, Berlin, 1990. Springer Verlag.

[108] B. B. Kimia, A. R. Tannenbaum, and S. W. Zucker. Entropy scale-space. In *Proceedings of the International Workshop on Visual Shapc*, Capri, Italy, May 1991. Plenum Press.

[109] B. B. Kimia, A. R. Tannenbaum, and S. W. Zucker. Entropy scale-space. In C. Arcelli, editor, *Visual Form: Analysis and Recognition*, pages 333–344, New York, May 1991. Plenum Press.

[110] B. B. Kimia, A. R. Tannenbaum, and S. W. Zucker. Nonlinear shape approximation via the entropy scale space. In *Proceedings of the SPIE's Geometric Methods in Computer Vision II*, volume 2031, pages 218–233, San Diego, California, July 1993.

[111] B. B. Kimia, A. R. Tannenbaum, and S. W. Zucker. On the shape triangle. In C. Arcelli, editor, *Proceedings of the International Workshop on Visual Form*, pages 307–323, Capri, Italy, May 1994. World Scientific.

[112] B. B. Kimia, A. R. Tannenbaum, and S. W. Zucker. Shapes, shocks, and deformations, I: The components of shape and the reaction-diffusion space. *IJCV*, 15:189–224, 1995.

[113] J. J. Koenderink. The structure of images. *Biological Cybernetics*, 50:363–370, 1984.

[114] J. J. Koenderink. *Solid Shape*. MIT Press, Cambridge, Massachusetts, 1990.

[115] J. J. Koenderink and A. J. van Doorn. Dynamic shape. *Biological Cybernetics*, 53:383–396, 1986.

[116] L. Lam, S.-W. Lee, and C. Y. Suen. Thinning methodologies-a comprehensive survey. *IEEE Trans. on PAMI*, 14(9):869–885, September 1992.

[117] L. J. Latecki and R. Lakamper. Convexity rule for shape decomposition based on discrete contour evolution. *Computer Vision and Image Understanding*, 73:441–454, 1999.

[118] Y. G. Leclerc. Constructing simple satble descriptions for image partitioning. *International Journal of Computer Vision*, 3:73–102, 1989.

[119] D. T. Lee. Medial axis transformation of a planar shape. *IEEE Trans. on PAMI*, 4(4):363–369, July 1982.

[120] R. J. LeVeque. *Numerical Methods for Conservation Laws*. Birkhauser Verlag, 1992.

[121] F. Leymarie and B. B. Kimia. Surface based euclidean distance transf rm (sedt). Technical report, LEMS, Brown University, April 1998.

[122] F. Leymarie and M. D. Levin . Fast raster scan distance propagation on the discrete rectangular lattic . *Computer Vision, Graphics and Image Processing*, 55(1):84–94, January 1992.

[123] F. Leymarie and M. D. Levine. Simulating the grassfire transform using an active contour model. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 14(1):56–75, January 1992.

[124] F. Leymarie. and M. D. Levine. Simulating the grassfire transform using an active contour model. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 14(1):56–75, 1992.

[125] F. Leymarie and M. D. Levine. Tracking deformable objects in the plane using an active contour model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 15:617–633, 1993.

[126] M. Leyton. A process grammar for shape. *Artificial Intelligence*, 34:213–247, 1988.

[127] T. Lindeberg. *Scale-space theory in computer vision*. Kluwer, 1994.

[128] P. Lions. *Generalized Solutions of Hamilton Jacobi Eqautions*. Pitman. 1981.

[129] J. D. Logan. *An Introduction to Nonlinear Diffential Equation*. John Wiles and Sons, Inc, 1994.

[130] D. Lowe. Organization of smooth image curves at multiple scales. *IJCV*. 3:119–130. 1989.

[131] D. G. Lowe. Organization of smooth image curves at multiple scales. In ICCV1988 [88], pages 558–567.

[132] R. Malladi and J. A. Sethian. Image processing: Flows under min/max curvature and mean curvature. *Computer Vision and Image Understanding*, 58:127–141, 1996.

[133] R. Malladi, J. A. Sethian, and B. C. Vemuri. Evolutionary fronts for topology-independent shape modelling and recovery. In *Proceedings of ECCV '94*, pages 3–13, 1994.

[134] R. Malladi, J. A. Sethian, and B. C. Vemuri. Shape modelling with front propagation: A level set approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 17, 1995.

[135] A. Manos and B. B. Kimia. Colored skeletons in scale. Technical Report Lems 135. Brown University, June 1994.

[136] D. Marr and E. Hildreth. Theory of edge detection. Technical Report MIT AI Memo 518. MIT AI Lab, 1979.

[137] G. Matheron. *Random Sets and Integral Geometry*. Wiley, 1975.

[138] F. Mokhtarian and A. Mackworth. Scale-based description of planar curves and two-dimensional shapes. *PAMI*, 8:34–43, 1986.

[139] F. Mokhtarian and A. Mackw rth. A theory of multiscale, curvature-based shape representation for planar curves. *PAMI*, 14(8):789–805, August 1992.

[140] U. Montanari. A method for obtaining skeletons using a quasi-euclidean distance. *JACM*, 15(4):600–624, October 1968.

[141] D. Mumford. Elastica and computer vision. In *Algebraic Geometry and Its Applications*, pages 491–506. Springer-Verlag, 1994.

[142] D. Mumford and J. Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on Pure and Applied Mathematics*, XLII:577–685, 1989.

[143] P. Neskovic and B. B. Kimia. Three-dimensional shape representation from curvature-dependent deformations. In *Proceedings of the IEEE International Conference on Image Processing*, pages 6–10. Austin, Texas, 1994. IEEE Computer Society Press.

[144] W. Niblack, P. B. Gibbons, and D. Capson. Generating connected skeletons for exact and approximate reconstruction. In *International Conference on Computer Vision and Pattern Recognition*, pages 826–828, 1992.

[145] M. Nitzberg, D. Mumford, and T. Shiota. *Filtering, Segmentation and Depth*. Springer-Verlag, 1993.

[146] W. Nuenschwander, P.Fua, and O. Kubler. Initializing snakes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 658–663, 1994.

[147] R. L. Ogniewicz. *Discrete Voronoi Skeletons*. Hartung-Gorre, 1993.

[148] R. L. Ogniewicz and M. Ilg. Voronoi skeletons: Theory and applications. In *CVPR92*, 1992.

[149] R. L. Ogniewicz and O. Kubler. Hierarchic voronoi skeletons. *Pattern Recognition*, 28(3):343–359, 1995.

[150] T. Ohya, M. Iri, and K. Murota. A fast voronoi-diagram algorithm with quaternary tree bucketing. *Information Processing Letters*, 18:227–231, 1984.

[151] J. Oliensis. Local reproducible smoothing without shrinkage. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(3):307–312, March 1993.

[152] S. Osher. Riemann solvers, the entropy condition, and difference approximations. *SIAM Journal of Numerical Analysis*, 21(2):217–235, 1984.

[153] S. Osher and L. I. Rudin. Feature-oriented image enhancement using shock filters. *SIAM Journal of Numerical Analysis*, 27(4):919–940, August 1990.

[154] S. Osher and J. A. Sethian. Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[155] S. Osher and C.-W. Shu. High-order essentially non-oscillatory schemes for Hamilton-Jacobi equations. *SIAM Journal of Numerical Analysis*, 28:907–922, 1991.

[156] P. Parent and S. W. Zucker. Trace inference, curvature consistency and curve detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 11(8):823–839, August 1989.

[157] T. Pavlidis. A thinning algorithm for discrete binary images. *Computer Graphics and Image Understanding*, 13:142–157, 1980.

[158] A. Pentland. Recognition by parts. In *First International Conference on Computer Vision, (London, England, June 8–11, 1987)*, Washington, DC., 1987. IEEE Computer Society Press.

[159] S. M. Pizer and C. A. Burbeck. Object representation by cores: Identifying and representing primitive spatial regions. *Vision Research*, 35(13):1917–1930, 1995.

[160] S. M. Pizer, W. R. Oliver, and S. H. Bloomberg. Hierarchical shape description via the multiresolution symmetric axis transform. *IEEE Trans. on PAMI*, 9(4):505–511, 1987.

[161] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, 1985.

[162] M. Proesmans, L. V. Gool, and A. Oosterlinck. Determination of optical flow and its discontinuities using non-linear diffusion. Technical Report KUL/ESAT/MI2/9308, Katolieke Universiteit Leuven, 1993.

[163] M. Proesmans, E. Pauwels, and L. V. Gool. Coupled geometry-driven diffusion equations for low-level vision. In *Geometry-Driven Diffusion in Computer Vision*. Kluwer, 1994.

[164] I. Ragnemalm. Generation of Euclidean distance maps. Licenciate Thesis No 206, Linkoping University, Sweden, 1990.

[165] I. Ragnemalm. Neighborhoods for distance transformations using ordered propagation. *CVGIP: Image Understanding*, 56(3):399–409, 1992.

[166] R. Ramamurthy and R. T. Farouki. Voronoi diagram and medial axis algorithm for planar domains with curved object boundaries i. Theoretical foundations. *Journal of Computational and Applied Mathematics*, to appear, 1998.

[167] R. Ramamurthy and R. T. Farouki. Voronoi diagram and medial axis algorithm for planar domains with curved object boundaries ii. Detailed algorithm description. *Journal of Computational and Applied Mathematics*, to appear, 1998.

[168] I. Rock. Gestalt psychology. *Scientific American*, ??, December 1990.

[169] A. Rosenfeld and A. C. Kak. *Digital Picture Processing*. Academic Press, 1982.

[170] A. Rosenfeld and J. Pflatz. Sequential operations in digital pictur processing. *Journal of the Association for Computing Machinery*, 13(4):471–494, October 1966.

[171] E. Rouy and A. Tourin. A viscosity solutions appr ach to shape-from-shading. *SIAM. Journal of Numerical Analysis*, 29(3):867–884, June 1992.

[172] R. Samadani. Adaptive snakes: control of damping and material parameters. In *Proc. SPIE Conf. Ge metric Methods Comput. Vision*, volume 1570, pages 202–213, 1991.

[173] G. Sanniti di Baja. Well-shaped, stable, and reversible skeletons from the (3-4)-distance transform. *Journal of Visiual Cummunication and Image Representation*, 5:107–115, 1994.

[174] G. Sapiro and A. Tannenbaum. On affine plane curve evolution. *Journal of Functional Analysis*, 119:79–120, 1994.

[175] S. Sarkar and K. Boyer. Perceptual organization using Bayesian networks. In *CVPR92*, pages 251–256, 1992.

[176] G. L. Scott, S. Turner, and A. Zisserman. Using a mixed wave/diffusion process to elicit the symmetry set. *Image and Vision Computing*, 7(1):63–70, February 1989.

[177] T. B. Sebastian, H. Tek, J. J. Crisco, and B. B. Kimia. Skeletally coupled deformable models. Technical Report LEMS 169, LEMS, Brown University, January 1998.

[178] T. B. Sebastian, H. Tek, S. W. Wolfe, J. J. Crisco, and B. K. Kimia. Segmentation of Carpal Bones from 3D CT images using Skeletally Coupled Deformable Models. In *International Conference on Medical Image Computing and Computer Assisted Interventions*, pages 1184–1194, Boston, MA, October 1998.

[179] J. Serra, editor. *Image Analysis and Mathematical Morpholpogy*. Academic Press, 1982.

[180] J. Serra, editor. *Image Analysis and Mathematical Morpholpogy, Part II: Theoretical Advances*. Academic Press, 1988.

[181] J. A. Sethian. An analysis of flame propagation. Ph.D. dissertation, University of California, Berekely, Berkeley, California, 1985.

[182] J. A. Sethian. Curvature and the evolution of fronts. *Comm. Math. Physics*, 101:487–499, 1985.

[183] J. A. Sethian. A review of recent numerical algorithms for hypersurfaces moving with curvature dependent speed. *J. Diff. Goem.*, (33):131–161, 1989.

[184] J. A. Sethian. Numerical algorithms for propagating interfaces: Hamilton-Jacobi equations and conservation laws. *J. Differential Geometry*, 31:131–161, 1990.

[185] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *Nat. Acad. Sci.*, 93:1591–1595, 1996.

[186] J. A. Sethian. *Level Set Methods*. Cambridge Univ rsity Press, New York, 1996.

[187] J. A. Sethian and J. Strain. Crystal growth and dendritic solidificati n. *Journal of Computational Physics*, 98:231–253, 1992.

[188] A. Sha'ashua and S. Ullman. Structural saliency: The detection of globally salient structures using a locally connected network. In ICCV1988 [88].

[189] J. Shah. A common framework for curve evolution, segmentation and anisotropic diffusion. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1996.

[190] D. Shaked and A. M. Bruckstein. Pruning medial axes. *Computer Vision and Image Understanding*, 69:156–169, 1998. 2.

[191] E. Sharon, A. Brandt, and R. Basri. Completion energies and scale. *IEEE Trans. Pattern Analysis and Machine Intelligence*, To appear.

[192] D. Sharvit, J. Chan, and B. B. Kimia. Symmetry-based indexing of image databases. In *Workshop on Content-Based Access of Image and Video Libraries. CVPR98*, pages 56–62, June 1998.

[193] C.-W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics*, 77:439–471, 1988.

[194] K. Siddiqi. Parts and shocks for recognition. Ph.D. dissertation, Division Of Engineering, Brown University, Providence, RI, 1995.

[195] K. Siddiqi and B. B. Kimia. Parts of visual form: Computational aspects. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(3):239–251, March 1995.

[196] K. Siddiqi and B. B. Kimia. A shock grammar for recognition. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 507–513, 1996.

[197] K. Siddiqi, B. B. Kimia, and C. Shu. Geometric shock-capturing ENO schemes for subpixel interpolation, computation and curve evolution. Technical Report 142, LEMS, Brown University, February 1995.

[198] K. Siddiqi, B. B. Kimia, and C. Shu. Geometric shock-capturing ENO schemes for subpixel interpolation, computation and curve evolution. *Graphical Models and Image Processing*, 59(5):278–301, September 1997.

[199] K. Siddiqi, A. Shokoufandeh, S. Dickinson, and S. Zucker. Shock graphs and shape matching. In *Proceedings of the Sixth International Conference on Computer Vision*, pages 222–229, Bombay, India, 1998. IEEE Computer Society Press.

[200] K. Siddiqi, K. J. Tresness, and B. B. Kimia. On the anatomy of visual form. In *Proceedings f the International Workshop on Visual Form*, pages 507–521, Capri, Italy, May 1994. World Scientific.

[201] K. Siddiqi, K. J. Tresness, and B. B. Kimia. Parts of visual form: Ecological and psychophysical aspects. *Perception*, 25:399–424, 1996.

[202] S. Smith and J. Brady. SUSAN - a new approach to low level image processing. In *Int. Journal of Computer Vision*, 1997. in publication.

[203] J. Smoller. *Shock Waves and Reaction-Diffusion Equations*. Springer-Verlag, New York, 1983.

[204] V. Srinivasan, L. R. Nackman, J.-M. Tang, and S. N. Meshkat. Automatic mesh generation using the symmetric axis transform of polygonal domains. *IEEE Proceedings*, 80:1485–1501.

[205] L. H. Staib and J. S. Duncan. Boundary finding with parametrically deformable models. *PAMI*, 14:1061–1075, 1992.

[206] P. Stoll, C. Shu, and B. B. Kimia. Shock capturing numerical methods for viscosity solutions of certain PDEs in computer vision: the Godunov, Osher-Sethian and ENO schemes. Technical Report 132, LEMS, Brown University, May 1994.

[207] K. Sugihara and M. Iri. Construction of the voronoi diagram for one million generators in single-precision arithmetic. *Proceedings of IEEE, Special Issue on Computational Geometry*, 80(9):1471–1484, 1992.

[208] t. McInerney and D. Terzopoulos. Topologically adaptible snakes. In *ICCV'95*, pages 840–845, 1995.

[209] S. Tari and J. Shah. Extraction of shape skeletons from grayscale images. *Computer Vision Image Understanding*, 66(2):133–146, 1997.

[210] S. Tari, J. Shah, and H. Pie. A computationally efficient shape analysis via level sets. In *IEEE Workshop on Mathematical Methods in Biomedical Image Analysis*, 1996.

[211] H. Tek and B. B. Kimia. Automatic volumetric segmentation of three-dimensional medical images. In *Proceedings of the Computer Assisted Radiology, 9th International Symposium and Exhibition*, Berlin, Germany, June 21–24 1995. Society for Computer Applications in Radiology.

[212] H. Tek and B. B. Kimia. Image segmentation by reaction-diffusion bubbles. In ICCV95 [90], pages 156–162.

[213] H. Tek and B. B. Kimia. Volumetric segmentation of medical images by three-dimensional bubbles. In *Proceedings of the IEEE Workshop on Physics-based Modelling in Computer Vision*, pages 9–16, Boston, Massachusetts, June 1995. IEEE Computer Society.

[214] H. Tek and B. B. Kimia. Volumetric segmentation of medical images by three-dimensional bubbles. *C mputer Vision and Image Understanding*, 64(2):246–258, February 1997.

[215] H. Tek and B. B. Kimia. Curve evolution, wave propagation, and mathematical m rphology. In *Fourth International Symposium on Mathematical Morphology (ISMM'98)*, pages 115–126, June 1998.

[216] H. Tek and B. B. Kimia. Symmetry map and symmetry transforms. In CVPR1999 [52], pages 471–477.

[217] H. Tek and B. B. Kimia. Symmetry maps of free-form curve segments via wave propagation. In ICCV1999 [89], pages 362–369.

[218] H. Tek, F. Leymarie, and B. B. Kimia. Multiple generation shock detection and labeling using CEDT. In *Proceedings of the International Workshop on Visual Form*, Capri, Italy, May 1997. World Scientific.

[219] H. Tek, P. Stoll, and B. B. Kimia. Shocks from images: Propagation of orientation elements. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 1997.

[220] B. M. ter Haar Romeny, editor. Kluwer, September 1994.

[221] B. M. ter Haar Romeny, L. Florack, J. Koenderink, and M. Viergever, editors. *Scale-Space Thoery in Computer Vision*. Springer, July 1997.

[222] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *Computer Graphics*, 21(4):205–214, 1987.

[223] D. Terzopoulos and A. Witkin. Constraints on deformable models: recovering shape and non-rigid motion. *Artificial Intelligence*, 36:91–123, 1988.

[224] K. Thornber and L. Williams. Analytic solution of stochastic completion fields. *Biological Cybernetics*, 75:141–151, 1996.

[225] S. Tirthapura, D. Sharvit, P. Klein, and B. B. Kimia. Indexing based on edit-distance matching of shape graphs. In *SPIE Inter. Symposium on Voice, Video, and Data Communications*, pages 25–36, November 1998.

[226] S. Ullman. Filling-in the gaps: The shape of subjective contours and a model for their generation. *Biological Cybernetics*, 25:1–6, 1976.

[227] B. Verwer. Improved metrics in image processing applied to the Hilditch skeleton. In *Ninth International Conference on Pattern Recognition (Rome, Italy, November 14–17, 1988)*, pages 137–142, Washington, DC, 1988. Computer Society Press.

[228] L. Vincent. Graphs and mathematical morphology. *Signal Processing*, 16(4):365–388, April 1989.

[229] L. Vincent. Exact euclidean distance function by chain propagations. In *Proceedings of the IEEE Computer Vision and Pattern Recognition '91*, pages 520–525, Maui HI, 1991.

[230] L. Vincent and P. Souille. Watersheds in digital spaces: an efficient algoritm based on immersi n simulations. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(6):583–598, 1991.

[231] J. W ick rt. Review of nonlinear diffusion filtering. In *First International Conference, Scale-Space*, pages 3–28, Utrecht, The Netherlands, 1997. Springer.

[232] R. Whitaker. Algorithms for implicit deformable models. In *International Conference on Computer Vision*, 1995.

[233] R. T. Whitaker. Geometry-limited diffusion in the characterization of geometric patches in images. *Computer Vision, Graphics and Image Processing*, 57(1):111–120, January 1993.

[234] L. Williams and D. Jacobs. Stochastic completion fields: A neural model of illusory contour shape and salience. In *Proceedings of the Fifth International Conference on Computer Vision*, pages 408–415, Boston, Massachusetts, June 1995. IEEE Computer Society Press.

[235] A. P. Witkin. Scale-space filtering. In *Proceedings of the* 8^{th} *International Joint Conference on Artificial Intelligence*, pages 1019–1022, Karlsruhe, West Germany, August 1983.

[236] H. Yamada. Complete Euclidean distance transformation by parallel operation. In *Proceedings of International Conference on Pattern Recognition*, pages 336–338, 1984.

[237] C. K. Yap. An O(n log n) algorithm for the voronoi diagram of a set of simple curve segments. *Discrete Computational Geometry*, 2:365–393, 1987.

[238] Q.-Z. Ye. The signed Euclidean distance transform and its applications. In *Proceedings of International Conference on Pattern Recognition*, pages 495–499, 1988.

[239] A. L. Yuille, P. W. Hallinan, and D. S. Cohen. Feature extraction from faces using deformable templates. *International Journal of Computer Vision*, 8(2):99–111, 1992.

[240] A. L. Yuille and T. A. Poggio. Scaling theorems for zero crossings. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(1):15–25, January 1986.

[241] S. C. Zhu and A. L. Yuille. Forms: A flexible object recognition and modeling system. *Int'l Journal of Computer Vision*, 20(3), 1996.

[242] S. C. Zhu and A. L. Yuille. Region competition: Unifying Snakes, Region growing, and Bayes/MDL for multiband Image Segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(9):884–900, 1996.

[243] S. W. Zucker. Region growing: Childhood and adolscence. *Computer Graphics and Image Processing*, 5:382–399, 1976.

[244] S. W. Zucker. *Early Vision*, pages 1131–1151. John Wiley and Sons, 1987.

[245] S. W. Zucker, A. Dobbins, and L. Iverson. Two stages of curve detection suggest two styles of visual computation. *Neural Computation*, 1:68–81, 1989.